Comandos Python: Strings

Bruno Musskopf

Agenda

Strings

- Quebra de linha (\n)
- Acesso aos caracteres da string com índices
- Operadores + e *
- Slicing
- Método strip
- Operador in
- Método find
- Método split
- Método replace
- Função list()
- Método join
- Método upper
- Método lower

Strings

- Strings em Python são listas imutáveis de caracteres.
- Strings são representadas por sequências de caracteres entre aspas simples ' ou entre aspas duplas ".

```
a = "20 de Abril tem prova."
a
'20 de Abril tem prova.'
b = 'Fizeram os exercícios?'
b
'Fizeram os exercícios?'
c = "Que vida \"fácil\""
c
'Que vida "fácil"'
```

Strings

- Strings em Python são listas imutáveis, portanto pode-se acessar posições de uma string de forma usual.

```
a = "20 de Abril tem prova."
print(a[0])
'2'
a[0] = "1"
```

```
---------------------------------------------------------------------------
TypeError Traceback (most recent call last)
<ipython-input-13-9ab1dda42293> in <module>()
----> 1 a[0] = "1"

TypeError: 'str' object does not support item assignment
```

Strings

- O caractere `'\n'` pode fazer parte de uma string e ele só causa a mudança de linha no comando **`print`**.

```
a = 'Fizeram\nos\nexercícios?'
print(a)
Fizeram
os
exercícios?
```

Strings: Operações, Funções e Métodos

- O operador + concatena 2 strings, e o operador* repete a concatenação (como em listas).

```
a = "20 de Abril tem prova."
b = 'Fizeram os exercícios?'
c = a + b
print(c)
'20 de Abril tem prova.Fizeram os exercícios?'
```

```
b = 'Fizeram os exercícios?\n'
print(3*b)
Fizeram os exercícios?
Fizeram os exercícios?
Fizeram os exercícios?
```

Strings como Listas

- Strings podem ser processadas como listas, podendo por exemplo ter seus elementos percorridos num laço **`for`**.
- Exemplo: Ler uma string e imprimir a inversa.

```
string = input("Digite um texto: ")
inversa = " "
for x in string:
    inversa = x + inversa
print(inversa)
```

Strings: Operações, Funções e Métodos

- A função **slice** (fatiar) devolve a string entre duas posições dadas.
- Pode-se fatiar (slice) strings usando `[inicio:fim-1:passo]`.

```
a = "20 de Abril tem prova."
print(a[6:11])
'Abril'
print(a[6:11:2])
'Arl'
print(a[::-1])
'.avorp met lirbA ed 02'
```

Strings: Operações, Funções e Métodos

- O método `strip` retorna uma string sem os brancos e mudança de linhas **no início e no final** de uma string.

```
b = "\nFizeram os exercícios?\n"
print(b)
'\n Fizeram os exercícios? \n'

print(b.strip(), “oi”)
'Fizeram os exercícios?'
```

Strings: Operações, Funções e Métodos

- O operador **in** verifica se uma **substring** é parte de uma outra string.

```
resultado = "exercícios" in "Fizeram os exercícios?"
print(resultado)
True

"ícios" in "Fizeram os exercícios?"
True

"Abril" in "Fizeram os exercícios?"
 False
```

Strings: Operações, Funções e Métodos

- O método `find` retorna onde a substring começa na string.

```
a = "Fizeram a atividade conceitual?"
print(a.find("atividade"))
10

print(a.find("abril"))
-1
```

- O método `find` retorna `-1` quando a substring não ocorre na string.

Strings: Operações, Funções e Métodos

- O método `split(sep)` separa uma string usando **sep** como separador. Retorna uma lista das substrings.

```
numeros = "1; 2 ; 3"
a = numeros.split(";")
print(a)
['1', ' 2 ', '  3']

b = "Fizeram a atividade conceitual?"
c = b.split()
print(c)
```

  - Podem haver substrings vazias no retorno de `split()`.

Strings: Operações, Funções e Métodos

- O método `replace` serve para trocar **todas** as ocorrências de
- uma substring por outra em uma string.

```
a = "Fizeram os exercícios?"

b = a.replace("exercícios","estudos dirigidos")

print(b)

a = "Fizeram a atividade conceitual?"
b = a.replace("conceitual","")

print(b)
'Fizeram a atividade ?'
```

Strings: Operações, Funções e Métodos

- Podemos usar a função `list` para transformar uma string em uma lista onde os itens da lista correspondem aos caracteres da string.

```
numeros = "1; 2 ;3"
listanumeros = list(numeros)
print(listanumeros)
```

Strings: Operações, Funções e Métodos

- O método `join` recebe como parâmetro uma sequência ou lista, e retorna uma string com a concatenação dos elementos da sequência/lista.

```
l = list("atividade")

print("-".join(l))
```

Strings: Operações, Funções e Métodos

- O método `lower` (`upper`) converte todos os caracteres da string para caixa baixa (alta).

```
str = "Atividade"
str1 = str.upper()
str2 = str.lower()
print(str)
Atividade
print(str1)
ATIVIDADE
print(str2)
atividade
```

Exercícios

Exemplo: Contador de Palavras

- Faça um programa que conta o número de palavras em um texto.

Exemplo: Contador de Palavras

- Faça um programa que conta o número de palavras em um texto.
  - Primeiramente removemos do texto todos os sinais de pontuação.

```
texto = input("Digite um texto: ")
pontuacao = [".", ",", ":", ";", "!", "?"]

# remove os sinais de pontuação
for p in pontuacao:
    texto = texto.replace(p," ")
```

Exemplo: Contador de Palavras

- Faça um programa que conta o número de palavras em um texto.
  - Depois usamos a função `split` para separar as palavras.

```
texto = input("Digite um texto: ")
pontuacao = [".", ",", ":", ";", "!", "?"]

# remove os sinais de pontuação
for p in pontuacao:
    texto = texto.replace(p," ")

# split devolve lista com palavras como itens
numero_palavras = len(texto.split())
print("Número de palavras:", numero_palavras)
```

Exercício: Palíndromo

- Faça um programa que lê uma string e imprime “Palíndromo” caso a string seja um palíndromo e “Não é palíndromo” caso não seja.
  - Assuma que a entrada não tem acentos e que todas as letras são minúsculas.

- Obs: Um *palíndromo* é uma palavra ou frase, que é igual quando lida da esquerda para a direita ou da direita para a esquerda (espaços em brancos são descartados).
  - Exemplos de palíndromo: “ovo”, “reviver”, “mega bobagem”, “anotaram a data da maratona”

Créditos

Os *slides* deste curso foram baseados nos slides produzidos e cedidos gentilmente pela Professora Sandra Ávila, do Instituto de Computação da Unicamp.