Manipulação de Strings

Vanessa Braganholo
vanessa@ic.uff.br

Strings

- Representam informação textual

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

Acesso a conteúdo das Strings

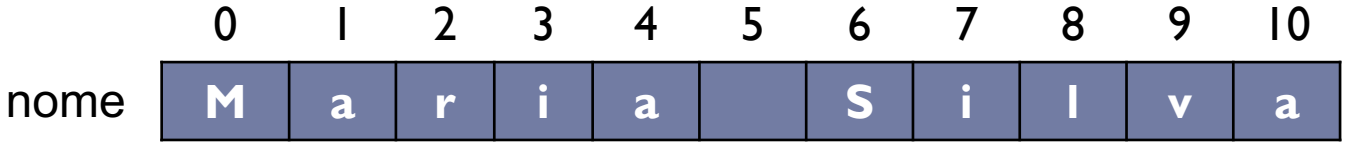
- Acesso pode ser feito pelo nome da variável que contém a string

```
nome = "Maria Silva"
print(nome)
```

Acesso a conteúdo das Strings

- String pode ser tratada como uma lista
- Caracteres podem ser acessados pela sua posição dentro da String

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



Alteração

- Diferentemente das listas, o conteúdo das strings 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
```

Operadores

- Alguns operadores que atuam sobre sequências podem ser usados em strings
  - in
  - len
  - +
  - *

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
```

len

- len(string)
  - Retorna a quantidade de caracteres da string

```
>>> nome = “Maria”
>>> len(nome)
5
>>> nome = “Maria Silva”
>>> len(nome)
11
```

+ (Concatenação)

- **string1 + string2**
  - Concatena duas strings

```
>>> nome = “Maria” + “Silva”
>>> nome
MariaSilva

>>> nome = “Maria”
>>> sobrenome = “Silva”
>>> nome_completo = nome + sobrenome
>>> nome_completo
MariaSilva
```

* (Repetição)

- string * int
  - Repete a string **int** vezes

```
>>> nome = “Maria”
>>> nome_repetido = nome * 2
>>> nome_repetido
MariaMaria
```

Percorrendo uma String

- Os elementos de uma string podem ser acessados usando uma estrutura de repetição

```
nome = "Maria Silva"
for letra in nome:
        print(letra)
```

```
nome = "Maria Silva"
for i in range(len(nome)):
        print(nome[i])
```

```
nome = "Maria Silva"
indice = 0
while indice < len(nome):
        print(nome[indice])
        indice +=1
```

Operações úteis sobre Strings

- upper
- lower

upper

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

- A string original não é modificada!

upper

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

lower

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

- A string original não é modificada!

lower

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

Exercícios

1. Escreva uma função que recebe uma frase e uma palavra antiga e uma palavra nova. A função deve retornar uma string contendo a frase original, mas com a última ocorrência da palavra antiga substituída pela palavra nova. A entrada e saída de dados deve ser feita no programa principal.

- Exemplo:
  - Frase: "Quem parte e reparte fica com a maior parte"
  - Palavra antiga: "parte"
  - Palavra nova: "parcela"
  - Resultado a ser impresso no programa principal: "Quem parte e reparte fica com a maior parcela"

Exercícios

2. Faça uma função que recebe uma string que representa uma cadeia de DNA e gera a cadeia complementar. A entrada e saída de dados deve ser feita pelo programa principal.

- Exemplo:
  - Entrada: AATCTGCAC
  - Saída: TTAGACGTG

Exercícios

3. Faça uma função que recebe uma frase e retorna o número de palavras que a frase contém. Considere que a palavra pode começar e/ou terminar por espaços. A entrada e saída de dados deve ser feita no programa principal.

4. Faça uma função que recebe uma frase e substitui todas as ocorrências de espaço por "#". Faça também uma função para realizar a entrada de dados. A saída de dados deve ser feita no programa principal.

Exercícios

5. Faça um programa que decida se duas strings lidas do teclado são palíndromas mútuas, ou seja, se uma é igual à outra quando lida de traz para frente.

Exemplo: **amor** e **roma**.

6. Um anagrama é uma palavra que é feita a partir da transposição das letras de outra palavra ou frase. Por exemplo, "Iracema" é um anagrama para "America". Escreva um programa que decida se uma string é um anagrama de outra string, ignorando os espaços em branco. O programa deve considerar maiúsculas e minúsculas como sendo caracteres iguais, ou seja, "a" = "A".

Exercícios

7. Faça um programa que leia o nome do usuário e mostre o nome de traz para frente, utilizando somente letras maiúsculas.

Exemplo:

Nome = Vanessa

Resultado gerado pelo programa:

ASSENAV

Exercícios

8. Faça um programa que leia o nome do usuário e o imprima na vertical, em forma de escada, usando apenas letras maiúsculas.

Exemplo:

Nome = Vanessa

Resultado gerado pelo programa:

V

VA

VAN

VANE

VANES

VANESS

VANESSA

Exercícios

9. Faça um programa que leia uma data de nascimento no formato dd/mm/aaaa e imprima a data com o mês escrito por extenso.

Exemplo:

Data = 20/02/1995

Resultado gerado pelo programa:

Você nasceu em 20 de fevereiro de 1995

Referências

- Slides de Aline Paes

Manipulação de Strings

Vanessa Braganholo
vanessa@ic.uff.br