[Aula 4-A] Linguagem de Programação

Análise e Desenvolvimento de Sistemas

Manipulação de Strings – Prof. Jean Zahn

jeanozahn@gmail.com

Strings

- Representam informação textual

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

Acesso a conteúdo das Strings

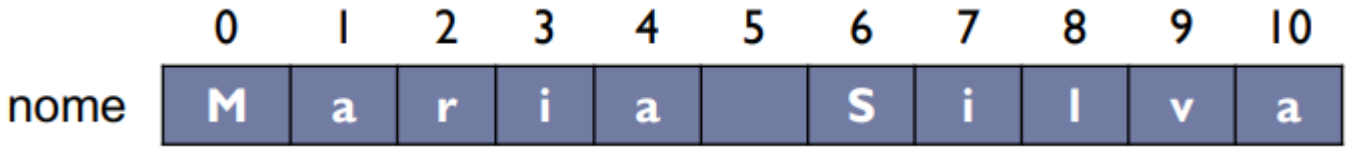
- Acesso pode ser feito pelo nome da variável que contém a String

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

Acesso a conteúdo das Strings

- 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

- 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
```

Operadores

- Alguns operadores importantes que 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"
i = 0
while i < len(nome):
        print(nome[i])
        i +=1
```

Percorrendo uma String

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

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

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

Percorrendo uma String

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

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

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

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

Operações úteis sobre Strings

- upper
- lower
- split
- partition

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_up = texto.upper()
>>> texto_up
"QUEM PARTE E REPARTE, FICA COM A MAIOR PARTE"
>>> texto
"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”

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 (no URI)

split

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

split

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

Pode-se usar um for para iterar

```
fila = input("Digite uma sequencia de números
separados por espaço: ")

for item in fila.split(" "):
    print(item)
```

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

partition

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

partition

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


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

Usando um while para iterar

```
fila = input("")

item, sep, fila = fila.partition(" ")

#condição de parada é quando um item da fila for igual
a zero
while int(item) != 0:
    print(item)
    #pega próximo item da fila
    item, sep, fila = fila.partition(" ")
```

Exercícios

1. Escreva um programa que lê uma frase e uma String antiga e uma String nova. O programa deve imprimir uma String contendo a frase original, mas com a última ocorrência da String antiga substituída pela String nova.

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

Exercícios

2. Faça um programa que lê uma String que representa uma cadeia de DNA e gera a cadeia complementar.

- Exemplo:
  - Entrada: AATCTGCAC
  - Saída: TTAGACGTG

Exercícios

3. Faça um programa que lê uma frase e retorna o número de palavras que a frase contém. Considere que a palavra pode começar e/ou terminar por espaços.

4. Faça um programa que lê uma frase e substitui todas as ocorrências de espaço por "#".

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 = Swainstainger

   Resultado gerado pelo programa:

   REGNIATSNIAWS

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

Agradecimentos

- Slides baseados no material de Aline Paes – Instituto de Computação – Universidade Federal Fluminense

[Aula 4-A] Linguagem de Programação

Análise e Desenvolvimento de Sistemas

Manipulação de Strings – Prof. Jean Zahn

jeanozahn@gmail.com