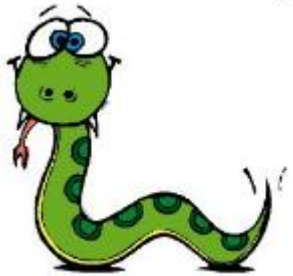
UFRPE
Universidade Federal Rural de Pernambuco



BSI
Sistemas de Informação
UFRPE

Manipulação de Strings

Prof. Wylliams Barbosa Santos
wylliamss@gmail.com
Introdução à Programação

Crédito de Conteúdo: Professora Ceça Moraes

Agenda

- String
  - Conceitos
  - Operações
  - Métodos
  - Exemplos
  - Exercícios

String

- S*trings* no Python são **elementos** usados para **armazenar texto**
- A **inicialização** de *strings* pode ser com aspas simples ou duplas
- **Imutáveis**
  - não é possível adicionar, remover ou mesmo modificar parte de uma *string*
  - Para realizar essas operações é necessário criar uma **nova *string***

Exemplos

```
>>> s = 'Camel'
```

- ```
  >>> print 'The ' + s + ' run away!'
  The Camel run away!
  ```
  ```

Exemplos


```
>>> s = 'Camel'
```

- **String como seqüência**

```
>>> for ch in s: print ch

C
a
m
e
l
```

- **Strings são objetos**

```
>>> if s.startswith('C'): print s.upper()

CAMEL
>>>
```

Exemplos


```
>>> s = 'Camel'
```

- **Tamanho de um string**

```
>>> len(s)
5
```

- **Pegando caracteres pelas suas posições**

```
>>> primeiraLetra = s[0]
>>> letraDoMeio = s[2]
>>> ultimaLetra = s[len(s)-1]
>>> print primeiraLetra,letraDoMeio, ultimaLetra
C m l
```

Exemplos


```
>>> s = 'Camel'
```

- o que acontecerá?

```
>>> print 3 * s
>>> # 3 * s é similar a s+s+s

>>> print 3 * s
CamelCamelCamel
>>> print s+s+s
CamelCamelCamel
>>>
```

Comparação de Strings


```
>>> palavra = "zebra"
>>> if palavra == "banana":print "Sim, nós não temos bananas!"
else: print "Não, nós não temos bananas!"

Não, nós não temos bananas!
```

```
>>> if palavra < "banana":
    print "Sua palavra," + palavra + ", vem antes de banana."
elif palavra > "banana":
    print "Sua palavra," + palavra + ", vem depois de banana."
else:
    print "Sim, nós não temos bananas!"

Sua palavra,zebra, vem depois de banana.
```

Atribuição

- Strings são imutáveis

```
>>> saudacao = "Alô, mundo!"
saudacao[0] = 'E'                    #  ERRO!
print saudacao
```

```
Alô, mundo!
```

```
>>> novaSaudacao = 'E' + saudacao[1:]
>>> print novaSaudacao
Elô, mundo!
```

Concatenação


```
>>> print 'Alô ''Mundo'
Alô Mundo
>>> print 'Alô ' + 'Mundo'
Alô Mundo
>>> print 'Alô! ' * 2 + 'Mundo'
Alô! Alô! Mundo
```

Interpolação

- Operador **%** é usado para fazer **interpolação** de *strings*

- **Mais eficiente** do que a **concatenação** convencional

```
>>> print 'Agora são %02d:%02d.' % (16, 30)

Agora são 16:30.
>>>
```

```
>>> s = 'Camel'

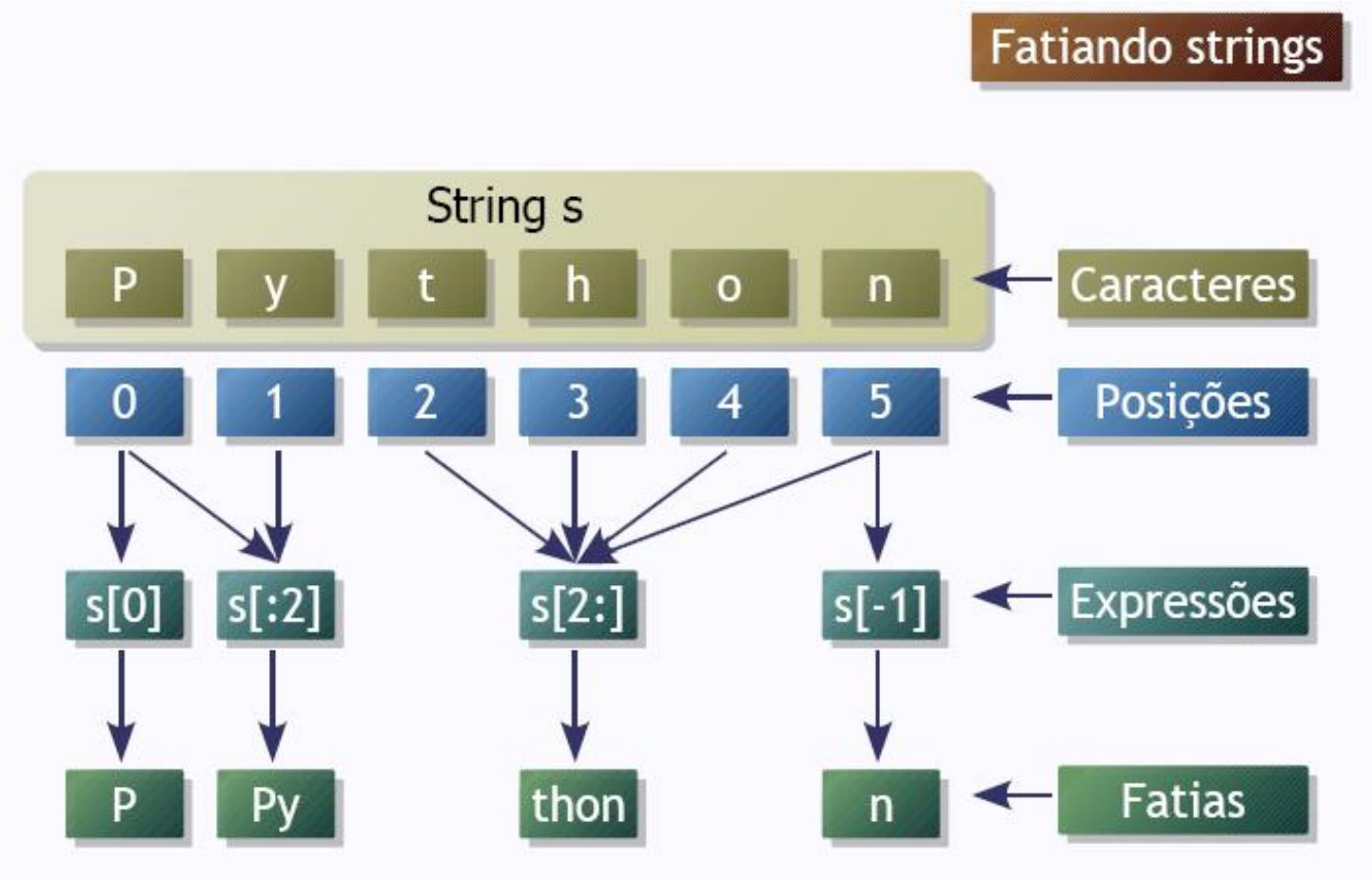
>>> print 'tamanho de %s => %d' % (s, len(s))
tamanho de Camel => 5
```

Interpolação

- Símbolos:
  - `%s`: *string*
  - `%d`: inteiro
  - `%f`: real

“Fatiando” Strings

- **`Fatias`** (***slices***) de *strings* podem ser obtidas colocando índices entre colchetes após a **variável** da *string*



Índices

- **Começam em zero**

- Podem ser definidos como trechos ou substrings:
  - `x[inicio:fim+1:intervalo]`
  - Se **não** for **definido** o `inicio`, será considerado como **zero**
  - Se **não** for **definido** o `fim+1`, será considerado **o tamanho** do objeto.
  - O `intervalo` (entre os caracteres), se não for definido, será **1**.

Índices


```
>>> x = 'multidisciplinar'
```

- Exemplos

```
>>> x[5:15:1]
'disciplina'

>>> x[5:15:3]
'dcla'

>>> x[5::1]
'disciplinar'

>>> x[:5]
'multi'
```

```
>>> len(x)
16
>>> x[0:16:5]
'mdpr'
>>> x[::5]
'mdpr'
```

Método `find`

- **`find (substring,inicio,fim)`**
  - Retorna o índice (**posição**) da primeira ocorrência de **`substring`**
  - **`inicio`** e **`fim`** são opcionais e indicam os intervalos de índices onde a busca será efetuada
    - Os defaults são **0** e o **comprimento** da string, respectivamente
  - Caso **`substring`** não apareça na string, é retornado o valor **-1**
  - Observe que o operador **`in`** pode ser usado para dizer se uma substring aparece numa string

Exemplo `find`


```
>>> s = "quem parte e reparte, fica com a maior parte“

>>> s.find("parte")
5

>>> s.find("reparte")
13

>>> s.find("parcela")
-1

>>> "parte" in s
True

>>> s.find("parte",6)
15

>>> s.find("parte",6,12)
-1
```

Método join

- `join(seqüência)`
  - Retorna uma string com todos os elementos da *seqüência* **concatenados**
  - Os elementos da sequência devem ser **strings**
  - A string objeto é usada como **separador** entre os elementos

Exemplos `join`


```
>>> separador = "/"
>>> separador.join(("23","11","2003"))
'23/11/2003'
```

Métodos
`lower` e `upper`

- **`lower()`**
  - Retorna a string com todos os caracteres convertidos para **minúsculos**

- **`upper()`**
  - Retorna a string com todos os caracteres convertidos para **maiúsculos**

- Exemplos:

```
>>> print "Casa".upper()
CASA
>>> print "MESA".lower()
mesa
```

Método `replace`

- **`replace(velho,novo,n)`**
  - Substitui as instâncias da substring **`velho`** por **`novo`**
  - Se **`n`** for especificado, apenas **`n` instâncias** são **trocadas**
  - Caso contrário, **todas** as instâncias são **trocadas**

Exemplo replace

```
>>> s = "quem parte e reparte, fica com a maior parte“

>>> s.replace("parte","parcela")
'quem parcela e reparcela, fica com a maior parcela‘


>>> s.replace("parte","parcela",2)
'quem parcela e reparcela, fica com a maior parte'
```

EXERCÍCIOS

Exercícios

1. Crie uma rotina que receba como argumento uma String (contendo uma palavra) e devolva o conteúdo dessa String invertido. Por exemplo, se o argumento recebido for “abcd”, a rotina deverá devolver “dcba”. Crie também um teste adequado.

Exercícios

2. Crie uma rotina que recebe como argumento uma String (contendo uma palavra ou frase) e devolve o conteúdo dessa String sem espaços (se existirem). Por exemplo, se o argumento recebido for " Universidade Federal Rural de Pernambuco ", a rotina deverá devolver “UniversidadeFederalRuraldePernambuco".

Exercícios

3. Crie uma rotina que recebe como argumento duas Strings, sendo que uma contém uma frase e outra contém uma expressão. O objetivo da rotina é criar uma nova frase em que qualquer ocorrência da expressão dada como segundo argumento é substituída por asteriscos ('*'). Por exemplo, se os argumentos recebidos forem "A minha senha é " e "12345", a rotina deverá devolver "A minha senha é *****".

Exercícios

4. Faça um programa que leia 2 strings e informe o conteúdo delas seguido do seu comprimento. Informe também se as duas strings possuem o mesmo comprimento e são iguais ou diferentes no conteúdo.

Exemplo:

```
Compara duas strings
String 1: Brasil Hexa 2006
String 2: Brasil! Hexa 2006!
Tamanho de "Brasil Hexa 2006": 16 caracteres
Tamanho de "Brasil! Hexa 2006!": 18 caracteres
As duas strings são de tamanhos diferentes.
As duas strings possuem conteúdo diferente.
```

Exercícios

5. Faça um programa que permita ao usuário digitar o seu nome e em seguida mostre o nome do usuário de trás para frente utilizando somente letras maiúsculas.

   Dica: lembre−se que ao informar o nome o usuário pode digitar letras maiúsculas ou minúsculas.

Exercícios

6. Faça um programa que solicite o nome do usuário e imprima-o na vertical.

Exemplo

```
F
U
L
A
N
O
```

Exercícios

7. Modifique o programa anterior de forma a mostrar o nome em formato de escada.

Exemplo

```
F
FU
FUL
FULA
FULAN
FULANO
```

Bibliografia

- Livro “Como pensar como um Cientista de Computação usando Python” – Capítulo 7
  - http://pensarpython.incubadora.fapesp.br/portal
- Python Tutorial
  - http://www.python.org/doc/current/tut/tut.html
- Dive into Python
  - http://www.diveintopython.org/
- Python Brasil
  - http://www.pythonbrasil.com.br/moin.cgi/DocumentacaoPython#head5a7ba2746c5191e7703830e02d0f5328346bcaac