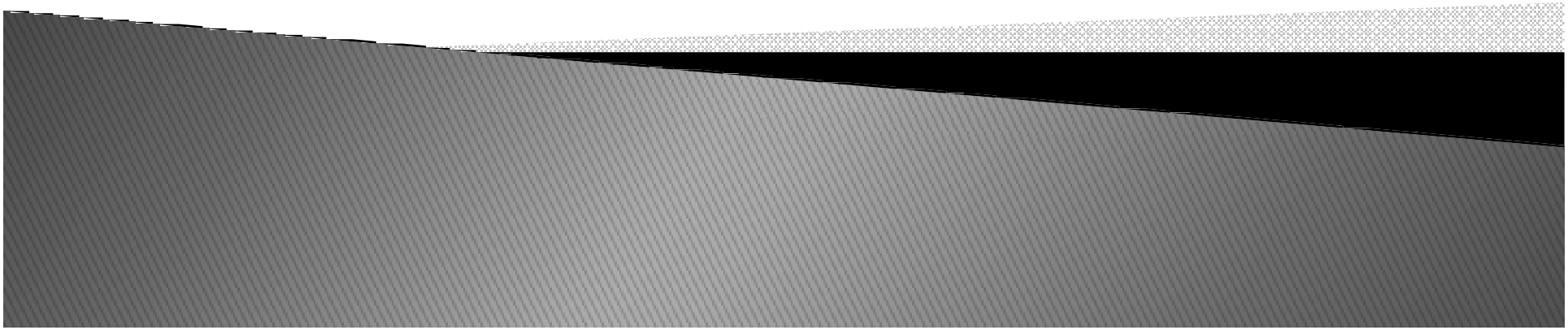
Introdução a Programação
aula05 – String

Bacharelado em Sistema de Informação

Prof. Gustavo Callou
gcallou@gmail.com



Tópicos

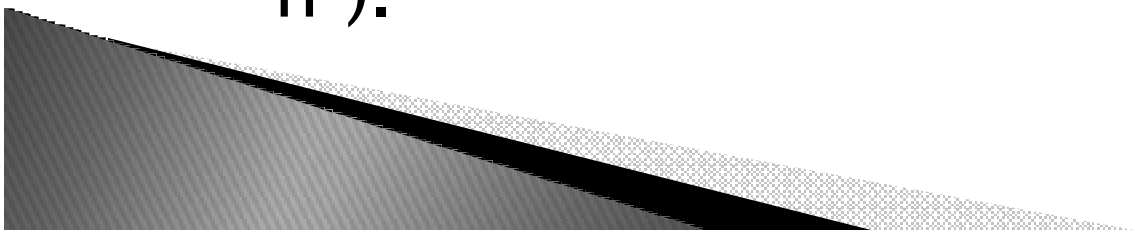
- String
  - conceitos
  - exercícios

Strings em Python

- As *strings* no Python são *buitins* para armazenar texto.
- São imutáveis→ não é possível adicionar, remover ou mesmo modificar algum caractere de uma *string*.
- Para realizar essas operações, o Python precisa criar um nova *string*.
- Tipos:
  - *String* padrão: s = 'Led Zeppelin'
  - *String unicode*: u = u'Björk'

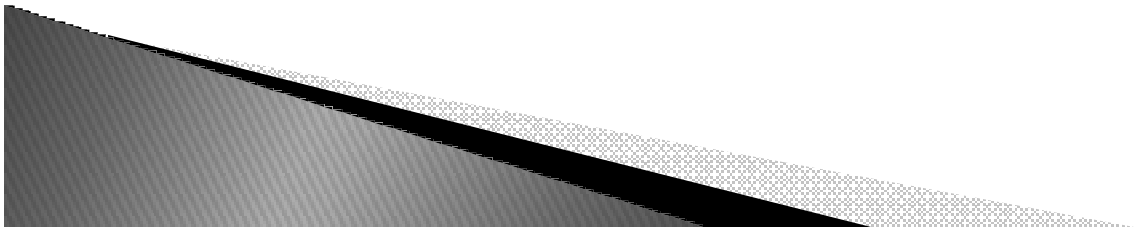
Strings em Python

- A *string* padrão pode ser convertida para *unicode* através da função *unicode()*.
- A inicialização de *strings* pode ser:
  - Com aspas simples ou duplas.
  - Em várias linhas consecutivas, desde que seja entre três aspas simples ou duplas.
  - Sem expansão de caracteres (exemplo: s = r'\n', aonde s conterá os caracteres “\” e “n”).



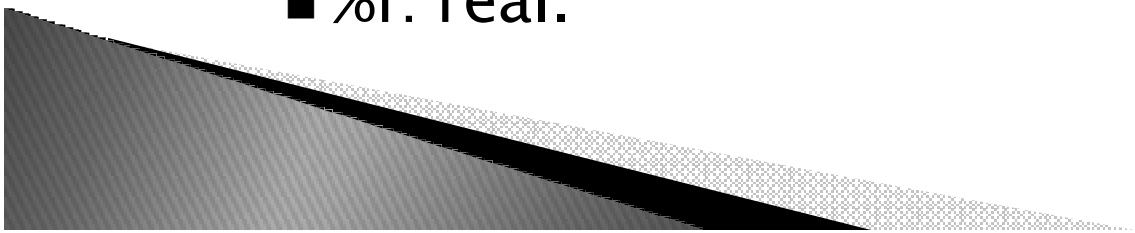
Exemplo

- s = 'Camel'
- # Concatenação
  - print 'The ' + s + ' run away!'
- # Interpolação
  - print 'tamanho de %s => %d' % (s, len(s))
- # String tratada como seqüência
  - for ch in s: print ch
- # Strings são objetos
  - if s.startswith('C'): print s.upper()
- # o que acontecerá?
  - print 3 * s
  - # 3 * s é consistente com s +s + s



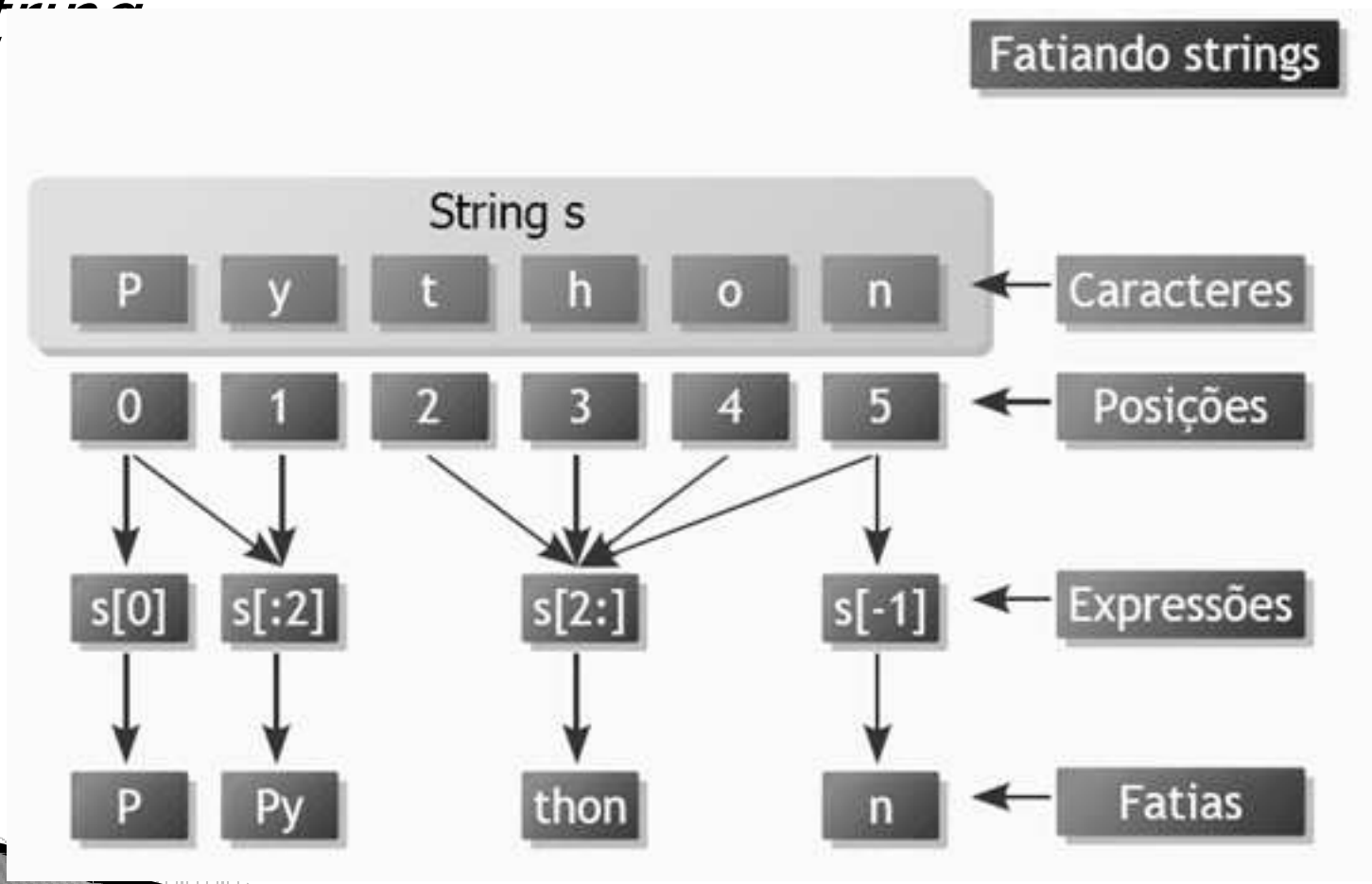
Exemplo

- Interpolação:
- Operador "%" é usado para fazer interpolação de *strings*. A interpolação é mais eficiente do que a concatenação convencional.
  - print 'Agora são %02d:%02d.' % (16, 30)
  - Símbolos usados na interpolação:
    - %s: *string*.
    - %d: inteiro.
    - %f: real.



Fatiando Strings

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

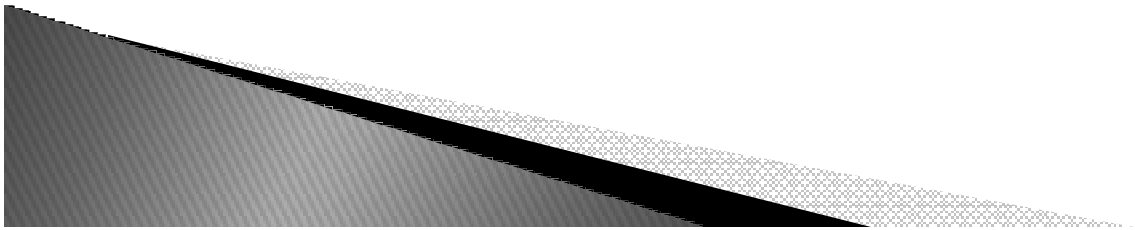


String

- Os índices no Python:
  - Começam em zero.
  - Contam a partir do fim se forem negativos.
  - Podem ser definidos como trechos, na forma [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.

Exercícios

- 1) Fazer um programa para contar quantos espaços em branco existe em uma frase.
-
- 2) Fazer um programa para contar quantos caracteres são diferentes de espaço em uma frase.
-
- 3) Fazer um programa para contar quantas vogais existem em uma frase.
- Dica: utilizar a idéia de conjuntos. Em pascal pode-se criar um conjunto e verificar se o elemento pertence ao conjunto, por exemplo: if 'a' in ['a','b','c','d'] then.....
- Ele verifica se o caractere 'a' pertence ao conjunto. Aqui também vale a idéia de seqüência igual no case. If 'a' in ['a' .. 'd'] then....
-
- 4) Fazer um programa para contar quantos números existem em uma frase.
-
- 5) Fazer um programa para concatenar duas strings lidas do usuários. A segunda string deve ser concatenada na primeira.
- Exemplo: string1: sol string2: lua
- após concatenar string1: sollua string2: lua
-



Exercícios

- 6) Fazer um programa semelhante ao anterior, mas agora as duas strings devem ser concatenadas e atribuídas a uma terceira string. Exibir o tamanho de todas elas após a concatenação.
- Exemplo: string1: sol string2: lua String3: sollua
- Tamanho: 3, 3, 6
-
- 7) Fazer um programa para ler uma frase e ver quantas vezes um determinado caractere aparece na frase. Esse caractere deve ser lido do usuários.
-
- 8) Fazer um programa para ler uma string e um caractere. Sempre que o caractere lido aparecer na frase ele deve ser substituído por asterisco.
- Exemplo: Frase: o dia esta nublado
- Caracter: d
- Resultado: o *ia esta nubla*o
-
- 9) Fazer um programa para ler uma frase e contar quantas palavras existem na frase.
-
- 10) Fazer um programa para ler uma frase e uma palavra. O programa deve verificar se a palavra existe na frase.