BCM0505-22 – Processamento da Informação

Strings

Carla Negri Lintzmayer
carla.negri@ufabc.edu.br
http://professor.ufabc.edu.br/~carla.negri/

Outline

Strings

Exemplos de problemas sobre strings

Exercícios

Strings

O que é uma String?

- Uma string é uma sequência de caracteres
  - Em Python, são objetos do tipo `str`
  - Em outras linguagens, são vetores de caracteres.
- Algumas coisas referentes a vetores também são aplicadas às strings em Python, como por exemplo a indexação e o tamanho.

```
1  nome = "Carla"
2  print(nome)
3  print(nome[0])
4  print(len(nome))
5
6  for i in range(len(nome)):
7      print(nome[i])
```

Strings em Python

- Apesar do acesso por indexação, strings em Python são **imutáveis**
- O seguinte código dá um erro de execução:

```
1  nome = "Carla"
2  nome[2] = "t"
```

Operações sobre Strings em Python

- Concatenação: `"exe" + "mplo"`
- Repetição: `"A" * 7`
- Slicing: `string[3:8]`
- Pertinência: `"a" in "aeiou"`

Métodos úteis sobre Strings em Python

```
1  texto = "aula de processamento da informação"
2  print(texto.split())
3  print(texto.split("a"))
4  print(texto.upper())
5  print(texto.lower())
6  print(texto.replace("aula", "Aula"))
7  print(texto.isalpha())
8  print(texto.isdigit())
9  print(texto.isascii())
```

Exemplos de problemas sobre strings

Palíndromos

Faça um programa que verifique se uma dada string é um palíndromo (a palavra é igual a ela mesma quando lida da direita para a esquerda).
Exemplos: “arara”, “radar”, “osso”.

Palíndromos

Faça um programa que verifique se uma dada string é um palíndromo (a palavra é igual a ela mesma quando lida da direita para a esquerda).
Exemplos: “arara”, “radar”, “osso”.

```
1  def eh_palindromo(palavra):
2      n = len(palavra)
3      for i in range(n):
4          if palavra[i] != palavra[n-i-1]:
5              return False
6      return True
```

Inversão

Faça um programa que recebe uma string e devolve-a invertida.

```
1  def inverte_str(texto):
2      invertida = ""
3      for i in range(len(texto)-1, -1, -1):
4          invertida += texto[i]
5      return invertida
```

Ordem lexicográfica

A ordem lexicográfica é a maneira como as palavras são ordenadas em um dicionário.
Por exemplo, “abacaxi” < “banana” < “carro” < “casamento” < “casar”.

Ordem lexicográfica

A ordem lexicográfica é a maneira como as palavras são ordenadas em um dicionário.
Por exemplo, “abacaxi” < “banana” < “carro” < “casamento” < “casar”.
Para comparar duas palavras:

- Primeiro comparamos o primeiro caractere das duas.
- Se forem diferentes, a palavra que tiver o caractere “menor” (alfabeticamente) vem primeiro.
- Se forem iguais, comparamos o segundo caractere, e assim por diante.
- Se uma palavra é um prefixo da outra (ex: “casa” e “casamento”), a menor é a mais curta.

Ordem lexicográfica

A ordem lexicográfica é a maneira como as palavras são ordenadas em um dicionário.
Por exemplo, “abacaxi” < “banana” < “carro” < “casamento” < “casar”.
Para comparar duas palavras:

- Primeiro comparamos o primeiro caractere das duas.
- Se forem diferentes, a palavra que tiver o caractere “menor” (alfabeticamente) vem primeiro.
- Se forem iguais, comparamos o segundo caractere, e assim por diante.
- Se uma palavra é um prefixo da outra (ex: “casa” e “casamento”), a menor é a mais curta.

Faça uma função que recebe duas palavras e devolve uma única string, com as palavras em ordem lexicográfica crescente e separadas por espaço. Se as palavras forem iguais, seu programa deve devolver uma delas apenas.

Ordem lexicográfica

```
1  def ordena_strs(palavra1, palavra2):
2      menor_tam = len(palavra1)
3      if len(palavra2) < len(palavra1):
4          menor_tam = len(palavra2)
5
6      # agora comparamos cada posição possível
7      for i in range(menor_tam):
8          if palavra1[i] < palavra2[i]:
9              return palavra1 + " " + palavra2
10         elif palavra2[i] < palavra1[i]:
11             return palavra2 + " " + palavra1
12
13     # se chegou aqui, os primeiros menor_tam caracteres são iguais
14     if len(palavra1) < len(palavra2):
15         return palavra1 + " " + palavra2
16     elif len(palavra2) < len(palavra1):
17         return palavra2 + " " + palavra1
18     else:
19         return palavra1
```

Ordem lexicográfica

Isso é bem mais fácil de fazer com Python, na verdade...

```
1  def ordena_strs(palavra1, palavra2):
2      if palavra1 < palavra2:
3          return palavra1 + " " + palavra2
4      elif palavra2 < palavra1:
5          return palavra2 + " " + palavra1
6      else:
7          return palavra1
```

Exercícios

Contar ocorrência

Faça uma função que recebe uma string e um caractere e devolve quantas vezes tal caractere aparece na string.

Por exemplo, se a função receber `Processamento da informacao` e o caractere `a`, deve devolver 4.

Eliminar espaços

Faça uma função que recebe uma string e devolve-a sem espaços.

Por exemplo, “olá mundo” deve virar “olámundo”.

Frases que são palíndromos

Algumas frases também podem ser consideradas palíndromos, quando ignoramos os espaços e pontuação.

Por exemplo, “Socorram-me, subi no ônibus em Marrocos” ou ”A sacada da casa”.

Faça uma função que recebe uma frase e verifica se ela é um palíndromo. Para simplificar, você pode considerar que as letras não terão acentos.

Anagramas

Faça uma função que recebe duas palavras e verifica se elas são anagramas (mesmas letras em ordens diferentes).

Por exemplo, “amor” e “roma”.

Formatação de texto

Faça uma função que recebe uma frase e coloca a primeira letra de cada palavra em maiúscula.

Validação de senhas

Faça uma função que verifica se um dado texto é uma senha válida.

Para ser uma senha válida, a string precisa ter 8 caracteres, pelo menos uma letra maiúscula, pelo menos um número e pelo menos um caractere ascii que não seja letra nem número (como, por exemplo, “-”, “_”, “@”, “#”).