Laboratório de Programação 1

Aula 09

Mário Hozano
professor@hozano.com

Ciência da Computação
UFAL - Arapiraca

Relembrando a aula anterior...

- Como acessar os caracteres de uma *string*?
- Para que serve a função *len() ?*
- O que são índices? Como trabalhar com índices negativos?
- O que são *substrings*?
- Como utilizar o operador *slice* (:)*?*
- É possível utilizar operadores relacionais com *strings*?

Roteiro da aula

- Listas
- Acessando os itens de uma Lista
- A função *len()* para listas
- Características de Mutação
- Métodos de Listas
- Removendo elementos de uma Lista
- Listas e *Strings*

Listas

- Assim como *strings*, uma lista também é uma sequência
- Neste caso, uma lista é uma sequência de valores que são chamados por items ou elementos
- A forma mais simples de criar uma lista é expressando os seus elementos entre colchetes e seperados por vírgulas
- Os elementos de uma lista podem ser acessados por índices assim como os caracteres de uma *string*

```
>>> lista = ['maria', 20, 1.75]
>>> lista[0]
'maria'
```

Listas

- Uma lista pode ter elementos de diferentes tipos

```
>>> lista1 = ['maria', 20, 1.75]
>>> lista2 = [13, 13.0, [13.12]]
>>> lista2 = [13, 'alow', False]
```

- Uma lista que não contém elementos é uma lista vazia e expressa por *[]*

```
>>> lista1 = []
```

Listas - Índices

- Assim como *strings* os elementos de uma lista são referenciados por índices iniciados pelo índice 0 (zero)
- Neste caso, o primeiro elemento de uma lista possui índice 0, o segundo possui índice 1, o terceiro índice 2...

```
>>> lista = [10, 'joao', 13.3, False]
>>> lista[0]
10
>>> lista[1]
'joao'
>>> lista[3]
False
```

Listas – A função *len()*

- Assim como em *strings* a função *len()* pode ser aplicada em listas
- Neste caso, a função retorna o número de elementos (tamanho) da lista dada

```
>>> lista = [10, 'joao', 13.3, False]
>>> len(lista)
4
>>> len([12, 22, 32])
3
```

Listas – Índices Negativos

- Os elementos das listas também podem ser acessados através de índices com valores negativos

- O índice -1 representa o último item, o índice -2 o penúltimo e assim sucessivamente

```
>>> lista = [10, 'joao', 13, False]
>>> lista[-1]
False
>>> lista[-2]
13
```

Listas – Características de mutação

- Diferentemente de *strings,* listas são mutáveis
- Uma vez criada é possível alterar o valor de um determinado elemento da lista
- A alteração do valor é feita com o comando de atribuição indicando o índice que deve ser alterado, como a seguir

```
>>> lista = [10, 'joao', 13, False]
>>> lista
[10, 'joao', 13, False]
>>> lista[2] = 13000
>>> lista
[10, 'joao', 13000, False]
```

Listas – Métodos de listas

- Como já sabemos, o método *append()* adiciona elementos ao final de uma lista

```
>>> lista = ['A', 'B', 'C']
>>> lista.append('D')
>>> lista
['A', 'B', 'C', 'D']
```

- Já o método *extend()* adiciona ao final da lista todos os elementos de uma outra lista dada como argumento

```
>>> lista = ['A', 'B', 'C']
>>> lista.extend(['D', 'E', 'F'])
>>> lista
['A', 'B', 'C', 'D', 'E', 'F']
```

Listas – Métodos de listas

- O método *sort()* organiza os elementos da lista deixando-os na ordem do menor para o maior

```
>>> lista = ['d', 'c', 'e', 'b', 'a']
>>> lista.sort()
>>> lista
['a', 'b', 'c', 'd', 'e']
```

- É importante observar que os métodos *append(), extend()* e *sort()* não retornam valores

- Estes métodos modificam apenas a lista na qual os métodos são chamados

Listas – Removendo Elementos

- Existem várias formas de remover elementos de uma lista
- Nos casos em que se sabe qual o índice do elemento que deve ser removido pode-se usar o método *pop()*
- O método *pop()* remove o elemento que possui o índice dado como argumento e o retorna na instrução

```
>>> lista = ['a', 'b', 'c', 'd', 'e']
>>> item = lista.pop(2)
>>> lista
['a', 'b', 'd', 'e']
>>> item
'c'
```

Listas – Removendo Elementos

- Uma outra forma de remover elementos de uma lista é usando a instrução *del*
- Ela deve ser usada também nos casos em que se sabe o índice do elemento a ser removido
- Diferentemente do método *pop()* a instrução não retorna nenhum valor

```
>>> lista = ['a', 'b', 'c', 'd', 'e']
>>> del lista[2]
>>> lista
['a', 'b', 'd', 'e']
```

Listas – Removendo Elementos

- Existem casos em que o programador necessita remover um elemento com um determinado valor que ele não sabe seu índice

- Nestes casos, pode ser usado o método *remove()* que exclui o elemento da lista que possui valor igual ao passado por argumento

```
>>> lista = ['a', 'b', 'c', 'd', 'e']
>>> lista.remove('c')
>>> lista
['a', 'b', 'd', 'e']
```

Listas e *Strings*

- Sabemos que uma *string* é uma sequência de caracteres e uma lista é uma sequência de valores

- Mas, uma lista de caracteres não é uma *string*

- Para transformar uma *string* em uma lista de caracteres deve-se usar a função *list()*

```
>>> s = 'UFAL'
>>> lista = list(s)
>>> lista
['U', 'F', 'A', 'L']
```

Listas e *Strings*

- Para dividir uma frase em uma lista de palavras pode-se utilizar o método *split() de strings*

- O método *split()* cria uma lista com as *substrings* que estão separadas por um espaço, como a seguir

```
>>> s = 'Universidade Federal de Alagoas'
>>> lista = s.split()
>>> lista
['Universidade', 'Federal', 'de', 'Alagoas']
```

Listas e *Strings*

- É possível ainda utilizar o método *split()* para criar uma lista com as *substrings* separadas por um delimitador qualquer

- Nestes casos o delimitador pode ser passado como argumento do método *split()*

```
>>> s = 'mario,ana,jose,joao'
>>> lista = s.split(',')
>>> lista
['mario', 'ana', 'jose', 'joao']
```

Exercícios

1. Crie uma função que receba duas listas de tamanho 10 com valores inteiros e retorne uma terceira lista com a soma dos elementos das listas passadas como argumento.

2. Escreva uma função que receba uma lista e remova as duplicatas da mesma.

3. Escreva uma função que recebe uma palavra e uma lista de palavras e retorne *True* se a palavra dada está contida na lista. *Falso* em outro caso.

4. Faça um programa que crie palavras a partir de uma lista de letras dada.