Processamento da Informação
– Teoria –

Coleções: Listas

Semana 06

Prof. Jesús P. Mena-Chalco

29/05/2013

Sobre strings ...

Podemos acessar (obter a informação) um caractere usando um **índice** (número inteiro).

A primeira letra é conseguida com o índice **zero**.

Valor **0** entre parênteses.

```
>>> x = “Joao Carlos”
>>> print x[0]
J
```

Sobre strings ...

Podemos acessar (obter a informação) um caractere usando um **índice** (número inteiro).

A primeira letra é conseguida com o índice **zero**.

Valor **0** entre parênteses.

```
>>> x = “Joao Carlos”
>>> print x[-1]
```

Sobre strings ...

Podemos acessar (obter a informação) um caractere usando um **índice** (número inteiro).

A primeira letra é conseguida com o índice **zero**.

Valor **0** entre parênteses.

```
>>> x = “Joao Carlos”
>>> print x[-1]
s
```

Sobre strings ...

Podemos acessar (obter a informação) um caractere usando um **índice** (número inteiro).

A primeira letra é conseguida com o índice **zero**.

Valor **0** entre parênteses.

```
>>> x = “Joao Carlos”
>>> print x[-2]
```

o

Sobre strings ...

Tamanho de uma string: len ← length

```
>>> x = “Joao Carlos”
>>> print len(x)
11
```

Sobre strings ...

Operador “:” para obter substrings

```
>>> x = “Joao Carlos”
>>> print x[2:4]                    ← Posições de 2 a 3
ao

>>> print x[5:11]                   ← Posições de 2 a 10
Carlos

>>> print x[5:]                     ← Posições de 2 até o final
Carlos

>>> print x[:4]                     ← Posições de 0 até o 3
Joao
```

Sobre strings ...

Strings são imutáveis

```
>>> x = “Joao Carlos”
>>> x[4] = “-”
```

Sobre strings ...

Strings são imutáveis

>>> x = “Joao Carlos”
>>> x[4] = “-”

```
def apagar_brancos(frase):
    k = 0
    while k<len(frase):
        if frase[k]==' ':
            frase[k]='-'
        k = k+1
    return frase
```

Listas (coleções)

Uma lista é uma coleção/sequência de elementos, onde cada elemento é identificado por um índice.

Uma lista é semelhante a uma String, com a diferença que é possível ter uma lista com qualquer tipo de elemento.

[10, 20, 30, 40]

['ufabc', 'ufmg', 'ufscar', 'ufrj']

Listas (coleções)

Uma lista é uma coleção/sequência de elementos, onde cada elemento é identificado por um índice.

Uma lista é semelhante a uma String, com a diferença que é possível ter uma lista com qualquer tipo de elemento.

[10, 20, 30, 40]

['ufabc', 'ufmg', 'ufscar', 'ufrj']

['ufabc', 10, 'ufmg', 20, 'ufscar', 30, 'ufrj', 40]

Os elementos da lista não necessitam ser do mesmo tipo

Listas (coleções)

A lista a seguir contém:
- uma string,
- um valor float,
- um valor inteiro,
- e um lista:

['spam', 2.0, 5, [10, 20]]

Uma lista dentro de outra lista é dita **estar aninhada**.

Listas (coleções)

Outro exemplo de listas aninhadas:

[[12], ['a', 'e', 'i', 'o', 'u'], [3.1415, 2.7182] **]**

Lista de listas

Função range (cria uma lista)

```
>>> range(1,5)
[1, 2, 3, 4]

>>> range(5,10)
[5, 6, 7, 8, 9]
```

Uma lista que não contém nenhum elemento é chamada de lista vazia.

```
>>> range(10,10)
[]
```

Criando uma lista

```
>>> nomes = ['Joao', 'Carlos', 'Maria']
>>> numeros = [17, 123]
>>> vazio = []
>>> var = [nomes, numeros, vazio]

>>> print var
[ ['Joao', 'Carlos', 'Maria'], [17, 123], [ ] ]
```

Listas **são mutáveis**

Para acessar um elemento da lista é a mesma sintaxe para acessar um caractere de um string.

```
>>> nomes = ['Joao', 'Carlos', 'Maria']

>>> print nomes[0]
Joao
>>> print nomes[2]
Maria

>>> print nomes[4]
erro de índice
```

Listas **são mutáveis**

O **operador colchete** pode aparecer em qualquer lugar em uma expressão.

Quando ele aparece no lado esquerdo de uma atribuição, ele modifica um dos elementos em uma lista.

```
>>> numeros = [17, 123]
>>> numeros[0] = [5]

>>> print numeros
[5, 123]
```

Operador '**in**'

O **operador in** também funciona com listas.

```
>>> nomes = ['Joao', 'Carlos', 'Maria']

>>> 'Carlos'  in nomes
1
>>> 'Carlo'  in nomes
0
```

Laço '**for**'

O laço **for** pode ser usado listas.

```
nomes = ['Joao', 'Carlos', 'Maria']
for nome in nomes:
    print nome
```

```
Joao
Carlos
Maria
```

Funciona bem para só para ler os elementos

Laço '**for**': exemplo

```
def soma_elementos(lista):
    resposta = 0
    for elementos in lista:
        resposta = resposta + elementos
    return resposta
```

```
>>> soma_elementos ([2,4,6,8])
20
>>> soma_elementos([pi, exp(1)])
5.8598
```

Laço 'for'

Para atualizar os items, é necessário percorrer usando um índice.

```
numeros = [1, 3, 6, 4]
for i in range(0, len(numeros)):
    numeros[i] = numeros[i] * 2
```

Dessa forma: numeros = [2, 6, 12, 8]

Elementos de uma lista

```
lista = ['sp', 1, ['1', '2', '2'], [1, 2, 3]]
```

Quantos itens contém a lista acima?

Elementos de uma lista

```
lista = ['sp', 1, ['1', '2', '2'], [1, 2, 3]]
```

Quantos itens contém a lista acima?

4

Concatenação de listas

```
>>> a = [ 1 , 3 , 5]
>>> b = [ 2 , 4 , 6]

>>> c = a+b

>>> print c
[1, 3, 5, 2, 4, 6]
```

Fatiamento de listas

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

>>> lista[1:3]
['b', 'c']

>>> lista[:4]
['a', 'b', 'c', 'd']
```

Fatiamento de listas

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

>>> lista[3:]
['d', 'e', 'f']

>>> lista[:]
['a', 'b', 'c', 'd', 'e', 'f']
```

Exercício 01

Crie uma função que permita contar o número de elementos em comum entre 2 listas dadas como parâmetro. Considere listas com elementos únicos.

Cabeçalho: def elementos_em_comum(L1, L2):

Exemplo:

L1= [1, 2, 3, 4, 5]
L2= [2, 4]
tem 2 elementos em comum.

L1= [1, 2, 3, 4, 5]
L2= [10]
tem **0** elementos em comum.

Exercício 01

```
def elementos_em_comum(L1, L2):
    contador = 0
    for elemento1 in L1:
        for elemento2 in L2:
            if elemento1==elemento2:
                contador += 1
    return contador
```

Exercício 01

```
>>> [1,2] == [2,1]
0

>>> [1,2] == [1,2]
1
```

Método **append**

O Python fornece também métodos que operam sobre listas.

Por exemplo, adicionar um novo elemento no final de uma lista

```
>>> t = ['a', 'b', 'c']
>>> t.append('d')
>>> print t

['a', 'b', 'c', 'd']
```

Exercício 02

Crie uma função que permita intercalar os elementos de duas listas de igual comprimento.

Cabeçalho: def intercala_listas(L1, L2):

Exemplo:

L1= [1, 3, 5]
L2= [2, 4, 6]
Resultado: [1,2,3,4,5,6]

L1= ['a','b','c','d']
L2= [10,20,30,40]
Resultado ['a', 10, 'b', 20, 'c', 30, 'd', 40]

Exercício 02

```
def intercala_listas(L1, L2):
    L3 = [ ]
    for i in range(0, len(L1)):
        L3.append(L1[i])
        L3.append(L2[i])
    return L3
```

```
>>> intercala_listas(['a','b','c','d'], [10,20,30,40 ])
['a', 10, 'b', 20, 'c', 30, 'd', 40]
```

Método **sort**

O método **sort** ordena a lista em ordem crescente.

```
>>> t = ['d', 'c', 'e', 'b', 'a']
>>> t.sort()
>>> print t

['a', 'b', 'c', 'd', 'e']
```

Exercício 03

Crie uma função que permita **somar todos os elementos** de uma lista.

Cabeçalho: def somar_elementos(lista):

Exemplo:

lista = [1, 3, 5]
Resultado: 9

lista = [10,20,30,40]
Resultado: 100

Exercício 03

```
def somar_elementos(lista):
    soma = 0
    for elemento in lista:
        soma = soma + elemento
    return soma
```

Exercício 03

```
def somar_elementos(lista):
    soma = 0
    for elemento in lista:
        soma += elemento
    return soma
```

Função **sum**

Pythom oferece uma opção abreviada para a somar todos os elementos de uma lista:

```
>>> t = [1, 2, 3]
>>> sum(t)
6
```

Eliminando elementos

Existem várias formas de eliminar elementos de uma lista. Usaremos apenas a função **'del'** para essa tarefa.

```
>>> t = ['a', 'b', 'c']
>>> del(t[1])
>>> print t
['a', 'c']
```

Eliminando elementos

Para eliminar mais de uma elemento, você pode utilizar o del com um pedaço de índices:

```
>>> t = ['a', 'b', 'c', 'd', 'e', 'f']
>>> del ( t[1:5] )
>>> print t
['a', 'f']
```

Atividade em Aula

Questao 1: Indique a mensagem que apresentará a execução da seguintes função. Considere como parâmetro de entrada a lista [1,2,4,16,32,64,-128]

```
def funcao1(lista):
    temp1 = lista[0]
    temp2 = lista[len(lista)-1]
    for elemento in lista:
        if temp1>elemento:
            temp1 = elemento
        if temp2<elemento:
            temp2 = elemento
    print str(temp1) + " " + str(temp2)
```

Atividade em Aula

Questao 2: Indique a mensagem que apresentará a execução da seguintes função. Considere como parâmetro de entrada a lista [1,2,4,16,32,64,-128]

```
def funcao2(lista):
    temp1 = lista[0]
    temp2 = lista[0]
    for elemento in lista:
        if temp1>elemento:
            temp2 = temp1
            temp1 = elemento
    print str(temp1) + " " + str(temp2)
```

Atividade em Aula

Questao 3: Indique o resultado apresentará a execução da seguintes função.

Considere como parâmetros de entrada:

L1=[1,3,4]
L2=[-1,0,2,5,7,9,10].

O que faz a função?

```
def funcao3(L1, L2):
    n1 = len(L1)
    n2 = len(L2)
    i = 0
    j = 0
    L3 = list([])
    while i<n1 and j<n2:
        if L1[i]<L2[j]:
            L3.append(L1[i])
            i = i+1
        else:
            L3.append(L2[j])
            j = j+1
    while i<n1:
        L3.append(L1[i])
        i = i+1
    while j<n2:
        L3.append(L2[j])
        j = j+1
    return L3
```

Atividade em Aula

Questao 1: Indique a mensagem que apresentará a execução da seguintes função. Considere como parâmetro de entrada a lista [1,2,4,16,32,64,-128]

```
def funcao1(lista):
    temp1 = lista[0]
    temp2 = lista[len(lista)-1]
    for elemento in lista:
        if temp1>elemento:
            temp1 = elemento
        if temp2<elemento:
            temp2 = elemento
    print str(temp1) + " " + str(temp2)
```

Função que imprime menor elemento

Resposta:
-128 64

Atividade em Aula

Questao 2: Indique a mensagem que apresentará a execução da seguintes função. Considere como parâmetro de entrada a lista [1,2,4,16,32,64,-128]

```
def funcao2(lista):
    temp1 = lista[0]
    temp2 = lista[0]
    for elemento in lista:
        if temp1>elemento:
            temp2 = temp1
            temp1 = elemento
    print str(temp1) + " " + str(temp2)
```

Função que imprime o primeiro e o segundo menor elemento

Resposta:
-128 1

Atividade em Aula

Questao 3: Indique o resultado apresentará a execução da seguintes função.

L1=[1,3,4]
L2=[-1,0,2,5,7,9,10].

Resposta:
[-1, 0, 1, 2, 3, 4, 5, 7, 9, 10]

O que faz a função?
Intercala listas ordenadas

```
def funcao3(L1, L2):
    n1 = len(L1)
    n2 = len(L2)
    i = 0
    j = 0
    L3 = list([])
    while i<n1 and j<n2:
        if L1[i]<L2[j]:
            L3.append(L1[i])
            i = i+1
        else:
            L3.append(L2[j])
            j = j+1
    while i<n1:
        L3.append(L1[i])
        i = i+1
    while j<n2:
        L3.append(L2[j])
        j = j+1
    return L3
```