Programação I

Licenciatura em Engenharia Informática

2015–2016

Vitor Beires Nogueira

Escola de Ciências e Tecnologia
Universidade de Évora

Uma lista é uma sequência

Definição

Uma lista é uma sequência de valores (denominados por elementos).

Exemplo

- `[10, 20, 30, 40]`
- `['crunchy frog', 'ram bladder']`
- `['spam', 2.0, 5, [10, 20]]`
- `[]`

As listas são mutáveis

Exemplo (Alteração de uma lista)

```
>>> numbers = [17, 123]
>>> numbers[1] = 5
>>> print numbers
[17, 5]
```

Exemplo (Operador in)

```
>>> cheeses = ['Cheddar', 'Edam', 'Gouda']
>>> 'Edam' in cheeses
True
>>> 'Brie' in cheeses
False
```

Percorrer uma lista

Exemplo (in)

```
for cheese in cheeses:
    print(cheese)
```

Exemplo (utilizando indices)

```
for i in range(len(numbers)):
    numbers[i] = numbers[i] * 2
```

Exemplo (lista vazia)

```
empty = []
for x in empty:
    print('This never happens.')
```

Operações sobre listas

Operador +

```
>>> a = [1, 2, 3]
>>> b = [4, 5, 6]
>>> c = a+ b
>>> print(c)
    [1, 2, 3, 4, 5, 6]
```

Operador *

```
>>> [1, 2, 3] * 3
    [1, 2, 3, 1, 2, 3, 1, 2, 3]
```

Manipulando partes de listas

```
>>> t = ['a', 'b', 'c', 'd', 'e', 'f']
>>> t[1:3]
    ['b', 'c']
>>> t[:4]
>>> ['a', 'b', 'c', 'd']
>>> t[3:]
>>> ['d', 'e', 'f']
>>> t[:]
>>> ['a', 'b', 'c', 'd', 'e', 'f']
>>> t[1:3] = ['x', 'y']
>>> print(t)
>>> ['a', 'x', 'y', 'd', 'e', 'f']
```

Métodos de listas

Adicionar um elemento: append

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

Concatenar uma lista: extend

```
>>> t1 = ['a', 'b', 'c']
>>> t2 = ['d', 'e']
>>> t1.extend(t2)
>>> print(t1)
    ['a', 'b', 'c', 'd', 'e']
```

Estes métodos (existem outros . . .)

- são todos *void*
- alteram a lista

Map

Exemplo (Duplica)

```
def duplica(t):
        for i in range(len(t)):
                t[i] *= 2
```

Filter

Exemplo (Somente_positivos)

```
def somente_positivos(t):
        res = []
        for e in t:
                if e >= 0:
                        res.append(e)
        return res
```

Reduce

Exemplo (Soma_todos)

```
def soma_todos(t):
        total = 0
        for e in t:
                total += e
        return total
```

Removendo elementos das listas I

pop

```
>>> t = ['a', 'b', 'c']
>>> x = t.pop(1)
>>> print(t)
    ['a', 'c']
```

del

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

Removendo elementos das listas II

remove

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

Objectos e valores

Mesmo objecto

```
>>> a = 'banana'
>>> b = 'banana'
>>> a is b
    True
```

Mesmo valor mas objecto diferente

```
>>> a = [1, 2, 3]
>>> b = [1, 2, 3]
>>> a is b
    False
```

Aliasing

Exemplo

```
>>> a = [1, 2, 3]
>>> b = a
>>> b is a
    True
>>> b[0] = 17
>>> print(a)
    [17, 2, 3]
```

- Tendo em conta o exemplo acima, não é aconselhável usar *aliasing* com objectos mutáveis.

Listas como argumentos

- As listas são passadas para as funções por referência.
- Se uma função alterar uma lista que seja seu parâmetro, quem invocou a função irá ver tal alteração.

Exemplo

```
def delete_head(t):
        del t[0]
```

```
>>> letters = ['a', 'b', 'c']
>>> delete_head(letters)
>>> print(letters)
    ['b', 'c']
```

Utilizando métodos de listas

Exemplo

```
>>> t1 = [1, 2]
>>> t2 = t1.append(3)
>>> print(t1)
    [1, 2, 3]
>>> print(t2)
    None
>>> t3 = t1 + [3]
>>> print(t3)
    [1, 2, 3]
>>> t1 is t3
    False
```

Delete head (problemático)

Exemplo (Bad_delete_head)

```
def bad_delete_head(t):
        t = t[1:]
```

Problema

Considere que pretende ler uma matriz de inteiros de um ficheiro para uma lista de listas.
Por exemplo, se o conteúdo do ficheiro for

```
12 34 45
21 43 65
```

deveria obter a matriz

```
[[12, 34, 45], [21, 43, 65]]
```

```
def matriz_ficheiro_lista(ficheiro):
    fin = open(ficheiro)
    linhas = le_linhas(fin)
    matriz = converte_linhas_matriz(linhas)
    fin.close()
    return matriz

def converte_linhas_matriz(linhas):
    matriz = []
    for linha in linhas:
        matriz.extend([converte_string_lista(linha)])
    return matriz

def le_linhas(fin):
    linhas = []
    for linha in fin:
        linhas.append(linha.strip())
    return linhas
```

```
def converte_string_lista(s):
    lista = []
    index = 0
    tamanho = len(s)
    while(index < tamanho):
        proximo = proximo_espaco(index, s)
        lista.extend([int(s[index:proximo])])
        index = proximo + 1
    return lista

def proximo_espaco(posicao, s):
    proximo = posicao + 1
    while (proximo < len(s) and s[proximo] != ' '):
        proximo += 1
    return proximo
```

Debugging

1. A maioria dos métodos das listas modificam o argumento e retornam `None`
   - `t = t.sort()` Errado!
2. É aconselhável escolher um modo de fazer as "coisas" e mante-lo ao longo do desenvolvimento
   - `t.append(x)`
   - `t = t + [x]`
   - `t.append([x])` Errado!
   - `t = t.append(x)` Errado!
   - `t + [x]` Errado!
   - `t + x` Errado!
3. De modo a evitar o *aliasing* devemos fazer cópias
   - `orig = t[:]`
   - `t.sort()`