DEPARTAMENTO DE INFORMÁTICA PUC-RIO

Listas

DEPARTAMENTO DE INFORMÁTICA PUC-RIO

Procura de elementos

- ✓ Para procurar um valor em uma lista

  lista.index (valor, início, fim)

- ✓ Exemplos

  l = [-15, 31, 25, 7, 6, 43, 25, -107]

  l.index(25) → 2

  l.index(25, 3) → 6

  l.index(25, 3, 7) → 6

  l.index(13) → ValueError: 13 is not in list

Contagem de ocorrências

- ✓ Para contar as ocorrências de um valor em uma lista

```
lista.count(valor)
```

- ✓ Exemplos

```
l = [-15, 31, 25, 7, 6, 43, 25, -107]
l.count(25)
```
→ 2

```
l.count(13)
```
→ 0

Inversão de uma lista

- ✓ Para inverter os elementos de uma lista

```
lista. reverse()
```

- ✓ Exemplos

```
l = [-15, 31, 25, 7, 6, 43, 25, -107]
l.reverse()
print (l)
```
→ [-107, 25, 43, 6, 7, 25, 31, -15]

Ordenação de uma lista

- ✓ Para ordenar os elementos de uma lista

```
lista. sort()
```

- ✓ Exemplos

```
l1 = [-15, 31, 25, 7, 6, 43, 25, -107]
l2 = ['camila', 'maya', 'flash']
l3 = [12, 5, 'A']
l1.sort()                    → [-107, -15, 6, 7, 25, 25, 31, 43]
l2.sort(reverse=True)        → ['maya', 'flash', 'camila']
l3.sort()                    → TypeError: '<' not supported between
                               instances of 'str' and 'int'
```

5

Exercício

- ✓ Escrever uma função chamada *intersec* que retorne uma lista representando a intersecção das duas listas recebidas como parâmetros
- ✓ Exemplo

```
l1 = [1, 2, 3, 4, 5, 6, 7, 8]
l2 = [0, 2, -4, 5, 8, 10, 12]
                              → [2, 5, 8]
```

6

Algortimo

```
lintersec = vazia
para cada elemento da lista1
    se elemento pertencer à lista2
        adicionar elemento à lista lintersec
retornar lintersec
```

7

Solução

```
# Função
def intersec (l1, l2):
    lintersec = list()
    for el in l1:
        if el in l2:
            lintersec.append(el)
    return lintersec

# Teste da função
l1 = [1, 2, 3, 4, 5, 6, 7, 8]
l2 = [0, 2, -4, 5, 8, 10, 12]
print (intersec (l1, l2))
```

→ [2, 5, 8]

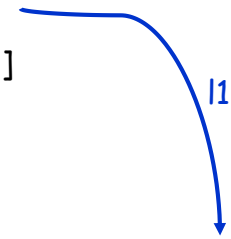
8

Exercício

- ✓ Escrever uma função chamada *elimina_dup* que receba duas listas (l1 e l2) e, utilizando a função *intersec*, remova de l1 todos os elementos comuns as duas listas
- ✓ Exemplo

l1 = [1, 2, 3, 4, 5, 6, 7, 8]

l2 = [0, 2, -4, 5, 8, 10, 12]



[1, 3, 4, 6, 7]

9

Algoritmo

```
para cada elemento da intersecção
    pos = posição do elemento em l1
    remover l1[pos]
retornar
```

10

DEPARTAMENTO DE INFORMÁTICA PUC-RIO

Solução

```
# Função
def elimina_dup (l1, l2):
    lintersec = intersec (l1, l2)
    for el in lintersec:
        pos = l1.index(el)
        l1.pop(pos)
    return
#
# Teste da função
#
l1 = [1, 2, 3, 4, 5, 6, 7, 8]
l2 = [0, 2, -4, 5, 8, 10, 12]
elimina_dupl (l1, l2)
print (l1)
```

[1, 3, 4, 6, 7]

11

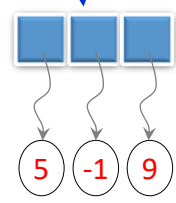
DEPARTAMENTO DE INFORMÁTICA PUC-RIO

Apelidos

- ✓ Uma lista pode ter um outro nome
  - apelido
  - os dois nomes referenciam o mesmo objeto lista
- ✓ Exemplo

```
l1 = [5, -1, 9]
l2 = l1
```

l1 e l2 referenciam a mesma lista

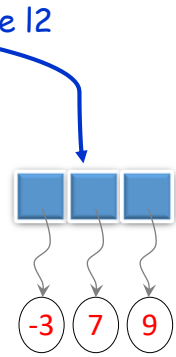


12

Apelidos

- ✓ Exemplo

```
l1 = [5, -1, 9]
l2 = l1
l1[0] = -3
l2[1] = 7
```

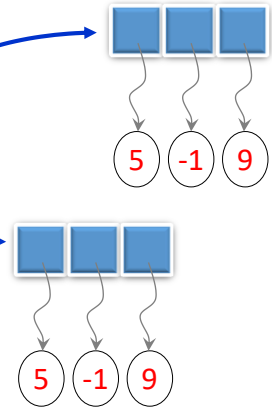


13

Clonagem

- ✓ Possível clonar uma lista
  - utiliza-se o operador de fatiamento
  - original e cópia referenciam objetos diferentes
  - *cópia*
- ✓ Exemplo

```
l1 = [5, -1, 9]
l2 = l1[:]
```



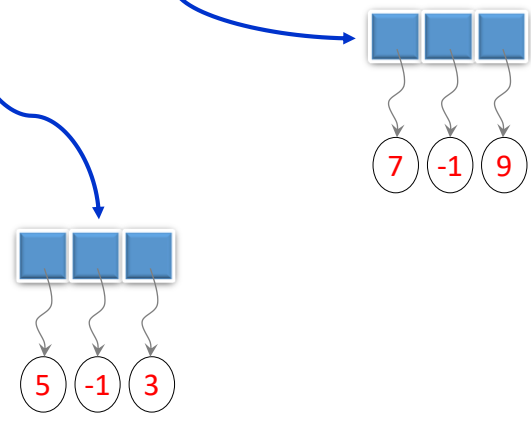
14

Clonagem

- ✓ Exemplo

l1[0] = 7

l2[2] = 3



15