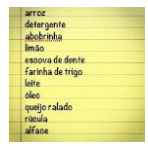
DEPARTAMENTO DE INFORMÁTICA PUC-RIO

Listas e Métodos

DEPARTAMENTO DE INFORMÁTICA PUC-RIO

Tipo Lista em Python

As listas têm tamanho variável. Crescem ou diminuem quando elementos são inseridos ou retirados



Lembrando que listas são **mutáveis**, ao contrário de strings. A qualquer momento, um item pode ser consultado e também pode:

- ✓ ser incluído na lista
- ✓ ser removido da lista
- ✓ ter seu valor alterado na mesma posição da lista.

Problema: inscrição de alunos

A coordenação da Informática deseja auxiliar a Secretaria a controlar as inscrições dos alunos em um curso de extensão. Só os alunos de Engenharia podem ser incluídos neste curso. A inscrição é presencial e realizada em um único dia.

Seu programa deve permitir a inclusão de novos alunos e a exclusão de alunos (já inscritos) que desistiram. Toda a operação inválida deve ser avisada.

Os pedidos são capturados via teclado do seguinte modo:

Operação desejada: "E" p/ excluir
"I" p/incluir
"F" p/finalizar

- ✓ Número de Matrícula: valor inteiro
- ✓ Curso: uma string com o nome

Ao final, mostrar as matrículas dos alunos inscritos no curso.

Solução para inscrição de alunos (1/7)

Como guardar a matrícula dos alunos inscritos?

1 lista de matrículas

Inicialmente, qual a situação desta lista?

[ ]

Solução para inscrição de alunos (2/7)

Pedidos de Inclusão (todos da engenharia)

161616, 162121, 1612222, 161212, 162121

[ ] 161616 ∉ lista de matrículas, inclui

[161616] 162121 ∉ lista de matrículas, inclui

[161616,162121] 161222 ∉ lista de matrículas, inclui

[161616,162121,162222] 161212 ∉ lista de matrículas, inclui

[161616,162121,162222,161212] 162121 ∈ lista de matrículas, Mensagem adequada

[161616,162121,162222,161212]

Solução para inscrição de alunos (3/7)

Inclusão quando é da Engenharia:

Se matrícula ∉ à lista de matrículas
    inclui matrícula
senão
    Envia mensagem

```
if matricula not in lMatr:
      lMatr+=[matricula]
else:
      print("Já incluído")
```

Solução para inscrição de alunos (4/7)

Pedidos de Exclusão (todos da engenharia)

162321, 162121

[161616, 162121, 162222, 161212] 162321 ∉ lista de matrículas, Mensagem adequada

[161616, 162121, 162222, 161212] 162121 ∈ lista de matrículas, retirar

↓

Encontrar seu índice na lista e concatenar as fatias antes e depois do eliminado

[161616, 162121, 162222, 161212]

[161616] + [162222, 161212]

[161616, 162222, 161212]

Solução para inscrição de alunos (5/7)

Exclusão quando é da Engenharia:

Se matrícula ∈ à lista de matrículas
    exclui matrícula
senão
    Envia mensagem

```
if matr in lMatr:
    ind = lMatr.index(matr)
    lMatr=lMatr[:ind]+lMatr[ind+1:]
else:
       print("Não existente")
```

Solução para inscrição de alunos (6/7)

Ler opção

Enq^{to} opção != "F" e opção != "f" faça

Ler matr e curso

Se o curso não for "Engenharia"

Enviar mensagem que o curso é só para Engenharia

Senão

Se opção for "I": incluir o aluno se $\notin$ à lista de inscritos

Senão: Se opção for "E": excluir o aluno se $\in$ à lista de inscritos

Ler opção

Solução para inscrição de alunos (7/7)

- ✓ Na inclusão a operação de concatenação cria uma cópia da lista com o novo elemento
- ✓ Na exclusão, a operação de fatiamento cria uma cópia da lista antecessora ao eliminado, uma cópia da lista sucessora ao eliminado e a operação de concatenação, cria uma cópia das listas geradas pelo fatiamento!!!!

Listas são sequências DINÂMICAS, mutáveis...
Como modificar a própria lista
incluindo um novo elemento?
retirando um elemento existente?

Inclusão no fim da lista: método *.append*

***lista*.append(*elemento*):** inclui *elemento* **no final** de *lista*

Exemplos: **print(lista)**

```
lista = ["a","b","c","d"]
lista.append("f")              ["a","b","c","d", "f"]
lista.append(["g","h"])        ["a","b","c","d","f",["g","h"]]
```

Inclusão em posição: método *.insert*

lista*.insert (*índice, elemento*):** inclui *elemento* antes do ***índice de *lista*

Exemplos: **print(lista)**

```
lista = ["b","d","e"]
lista.insert(0,"a")     ["a","b","d","e"]
lista.insert(2,"c")     ["a","b","c","d","e"]
lista.insert(-1,"f")    ["a","b","c","d","f","e"] # cuidado: antes do último
lista.insert(len(lista),"g")    ["a","b","c","d","f","e","g"] # último
lista.insert(20,"h")    ["a","b","c","d","f","e","g","h"] # último
lista.insert(1,["a","d"])   ["a",["a","d"],"b","c","d","f","e","g","h"]
                                                        #sl  na 2ª
```

Estendendo lista: método *.extend*

lista*.extend (*sequência**): adiciona elementos da *sequência* no final de *lista

* Qualquer objeto iterável

Exemplos: **print(lista1)**

```
lista1=["a","b"]
lista2 =["e","f"]

lista1.extend(lista2)    ["a", "b", "e", "f"]
lista1.extend("Oi")      ["a", "b", "e", "f", "O", "i"]
```

Mãos na massa!!
Teste e analise o resultado

```
A)
def difAppendExtend(lista):
    lista.append("tatu bola")
    lista.append(["tatu",1])
    print("Inclusão  com append ==> ",lista)

    print("Inclusão  com extend ")
    lista.extend(["tatu",1])
    print(".extend(["tatu",1])==>", lista)
    lista.extend("tatu bola")
    print(".extend("tatu bola")==> ", lista)

lista=[]
difAppendExtend(lista)

B)  Inclua na função acima e teste
    lista.append(1)
    lista.extend(1)
```

Resultados dos testes

A)

Inclusão com append ==> ["tatu bola", ["tatu", 1]]

Inclusão com extend

.extend(["tatu",1])==> ["tatu bola", ["tatu", 1], "tatu", 1]

.extend("tatu bola")==> ["tatu bola", ["tatu", 1], "tatu", 1, "t", "a", "t", "u", " ", "b", "o", "l", "a"]

B)

.append(1)==> ["tatu bola", ["tatu", 1], "tatu", 1, "t", "a", "t", "u", " ", "b", "o", "l", "a", 1]

lista.extend(1) TypeError: "int" object is not iterable

Exclusão de elemento: instrução *del*

del *lista*[indice]: remove o elemento. Um erro ocorre quando o índice está fora dos limites

del *lista*[a : b] : remove a sublista definida pela fatia

Exemplo 1: **print(lista)**

```
lista = ["a","b","c","d","e","f","g"]
del lista[4]        ["a","b","c","d","f","g"]
del lista[90]       IndexError: list assignment index out of range
del lista[1:3]      ["a","d","f","g"]
```

Exclusões usuais via Instrução *del*

Seja a *lista* abaixo:

lista = ["a","b","c","d","e","f","g"]

Exemplo 2: **print(lista)**

```
del lista[0]                ["b","c","d","e","f","g"]   #remove o 1º
del lista[-1]               ["b","c","d","e","f"]       # remove o último
del lista(len(lista)/2)     ["b","c","e","f"]           # remove o do meio
del lista[ : ]              [ ]                         # remove todos
del lista                   NameError:lista is not defined   #apaga lista
```

Exclusão em Listas método: *.pop*

***lista*.pop*()*:** exclui **o último** elemento de *lista*

lista.pop*(índice)*: exclui o elemento do ***índice*** de *lista*, retornando o valor removido. Um erro ocorre quando o índice está fora dos limites

Exemplos: **print(lista)** **print(el)**

```
lista =["a","b","c","d","e"]
el = lista.pop(   )          ["a","b","c","d"]     "e"
el = lista.pop(0)            ["b","c","d"]         "a"
lista.pop (90)               IndexError: pop index out of range
```

Exclusão em Listas: método *.remove*

***lista*.remove(*elemento*):** exclui a 1ª ocorrência de *elemento* da *lista*. Um erro ocorre quando o elemento não é encontrado.

Exemplos:

print(lista)

```
lista = ["b","d","e","b","d","e",["b","d","e"]]
lista.remove("b")    ["d","e","b","d","e",["b","d","e"]]
lista.remove("b")    ["d","e","d","e",["b","d","e"]]
lista.remove("b")  ValueError: list.remove("b"): "b" not in list
```

Mãos na massa!!
Teste e analise o resultado

```
def difDelPopRemove(lista):
    print(lista)
    del lista[0]
    print(lista)
    el= lista.pop()
    print(lista)
    lista.insert(0,el)
    print(lista)
    el = lista.pop(3)
    print(lista)
    lista.insert(len(lista),el)
    print(lista)
    lista.remove(8)
    print(lista)

lista=[1,2,3,4,5,6,7,8]
difDelPopRemove(lista)
```

Mãos na massa: Resultados!

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

del 1º ==> [2, 3, 4, 5, 6, 7, 8]

.pop()==> [2, 3, 4, 5, 6, 7]

inserção do retirado, como 1º [8, 2, 3, 4, 5, 6, 7]

.pop(3)==> [8, 2, 3, 5, 6, 7]

inserção do retirado, como último [8, 2, 3, 5, 6, 7, 4]

removeu o nº 8 ==> [2, 3, 5, 6, 7, 4]

Revisitando as inscrições em curso

Controle das inscrições dos alunos em um curso de extensão:

- Só p/aluno de Engenharia
- Operações possíveis
  - "E" p/ excluir
  - "I" p/incluir
  - "F" p/finalizar
- Cada aluno introduz:
  - Operação desejada
  - Matrícula:inteiro
  - Curso: string

Ao final, mostrar os alunos inscritos no curso.

Uma Solução possível

```
lMatr = []
op=input("Digite a operação: I-inclusão, E-exclusão, F-fim: ")

while op !="F" and op!="f":
    matr = int(input ( "Digite a matricula do aluno: "))
    curso = input ( "Digite o curso do aluno  "+ str(matr) +": ")
    if curso!= "engenharia" and curso != "Engenharia":
        print(matr, "você não pode se inscrever. Só para alunos de Engenharia")
    elif op =="i" or op =="I":
        incluiLista(lMatr,matr)
    elif op =="e" or op =="E":
        excluiLista(lMatr,matr)
    else:
        print("Operação inválida")
    print(lMatr)
    op=input("Digite a operação: I-inclusão, E-exclusão, F-fim:")
print("Alunos incluidos:", lMatr)
```

Faça as funções:
incluiLista
excluiLista

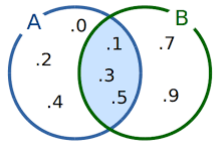
Funções *inclui* e *exclui* de lista

```
def incluiLista(l,matr):
    if matr in l:
        print(matr," já incluído")
    else:
        l.append(matr)
        print(matr,"incluído")
    return

def excluiLista(l,matr):
    if matr not in l:
        print(matr," não matriculado")
    else:
        l.remove(matr)
        print(matr,"excluído")
    return
```

Interseção de conjuntos

Faça uma função que receba dois conjuntos e retorne a interseção



$A \cap B = \{ x / x \in A \text{ e } x \in B \}$

Interseção = ∅
Para cada x de A
    Se x ∈B
        Adiciona x à Interseção

Uma Solução para Interseção

```
def intersecao(A,B):
    inters = []
    for el in A:
        if el in B:
          inters.append(el)

    return inters

A=[1,2,3,4,5]
B=[1,3,5,7,9,11]
print(intersecao(A,B))
```

Exercício: números repetidos

Faça uma função que receba uma lista com sequências de números repetidos deixando apenas 1 ocorrência de cada número.

DICA: encontre o índice do final da fatia de iguais e remove a fatia de iguais

Exemplo

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

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

Uma solução para num. repetidos

```
def encontraFatia(l,val):
    i=0
    while i<len(l) and l[i] == val:
        i+=1
    return i

l = [1,1,1,2,2,3,3,3,3,4,4,4,5,6,7,7,7,8,8]
for ind,el in enumerate(l):
    ult=encontraFatia(l[ind+1:],el)
    if ult!=0:
        del l[ind:ind+ult]  #remove as n 1ªs ocorrências, deixando a última
print(l)
```

Exercícios: rotação e particionamento

Crie funções que:

a) receba uma lista *l,* e retorne uma nova lista l*rot* onde cada posição está rotacionada em 2 posições, isto é, indice 0 equivale a indice + 2.

Dica: para calcular o índice em lrot, utilize o resto da divisão

Exemplo: *l = [1,2,3,4,5]* ➔ *lrot = [3,4,5,1,2]*

b) receba uma lista *l* e um número *max.* Esta função, retorna uma nova lista *lmax*, que particiona *l* em *max* elementos.

Exemplo: *l = [1,2,3,4,5]* *max* = 3 ➔ *lmax = [[1,2,3],[4,5]]*

Solução para Rotação

```
def  rotaciona(l):
    lrot=[]
    tam = len(l)
    for i in range(len(l)):
        lrot.append(l[(i+2)%tam])
    return lrot

l=[1,2,3,4,5]

print(rotaciona(l))
print(rotaciona([ ]))
print(rotaciona([1,2]))
```

Solução para Particionamento

```
def quebra(l,max):
    lmax=[]
    i=0
    while l[i:i+max]:
        lmax.append(l[i:i+max])
        i+=max
    return lmax


l=[1,2,3,4,5]
print(quebra(l,3))
print(quebra(l,9))
```

Ajustando notas dos alunos

Os 5 alunos de uma turma foram muito mal em uma prova.

O professor resolveu, então, considerar a maior nota como sendo nota 10.0 e transformar as demais notas em relação a esta nota da seguinte maneira: nota do aluno * 10/ maior nota.

Faça um programa que leia as matrículas e notas dos alunos, calcule a nova nota dos alunos, mostrando suas matrículas, as notas originais e as novas notas na tela.

		Matr	*Notas*	
			Ant	*Nova*
Exemplo:		1010	3.0	5.0 (3*10)/6
		1020	4.0	6.6
	maior nota: 6.0	1030	5.0	8.3
		1040	6.0	10.0
		1050	3.0	5.0

Solução: notas ajustadas (1/3)

Idéia da solução:

1. *Preencher a lista com [matricula,nota] lidos de cada aluno*
2. *Encontrar a maior nota nesta lista*
3. *Calcular e adicionar a cada aluno a nova nota*
4. *Mostrar na tela os dados dos alunos*

Criar uma lista vazia e adicionar no seu final a sublista [matr,nota] lida do arquivo

Solução: notas ajustadas (2/3)

```
def leAlunos():
    lista=[]
    for i in range(5):
        matr=int(input("Digite a matricula do aluno %d: "%(i+1)))
        nota=float(input("Digite a nota  do aluno %d: "%matr))
        lista.append([matr,nota])
    return lista

def maiorNota(lista):
    maior=-1
    for el in lista:
        if el[1]>maior:
            maior=el[1]
    return maior
```

Solução: notas ajustadas (3/3)

```
def ajustaNotas(l,maior):
    for el in l:
        el.append(el[1]*10/maior)
    return

def exibe(l):
    print("%12s %12s %12s"%("Matricula","Nota Antiga","Nota Nova"))
    for el in l:
        print("%12d %12.1f %12.1f"%(el[0],el[1],el[2]))
    return


lista=leAlunos()
maior=maiorNota(lista)
ajustaNotas(lista,maior)
exibe(lista)
```

Exercício: Sequências numéricas

Complete os espaços com os números que faltam. Preste bastante atenção para descobrir a lógica de cada fase

http://www.escolagames.com.br/jogos/completandoNumeros/

Desenvolvendo a Solução

1) Criar um sequência de números com uma lei de formação
2) Esconder um (ou mais) número(s) da sequência, incluindo-os em outra sequência.
3) Adivinha(sequência de números, sequência escondidos)

Desenvolvendo a Solução

PROBLEMA: Como criar a sequência com os números escondidos?

Como os números originais não podem ser "apagados" , a lista com alguns números escondidos deve ser criada a partir da lista original

SOLUÇÃO POSSÍVEL: Criar uma lista com mesmos valores da sequência de números (cópia) e apagar os elementos desejados na cópia.

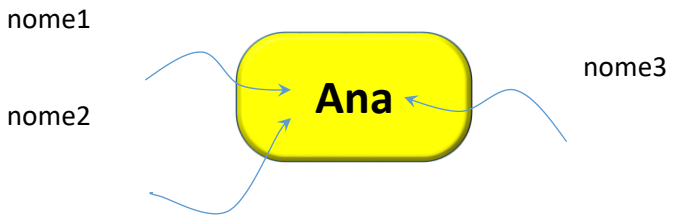
PROBLEMA: O comando de atribuição NÃO cria uma cópia!

Copiar x Clonar (1/8)

nome1 = "Ana"

nome2 = "Ana"

nome3 = nome1

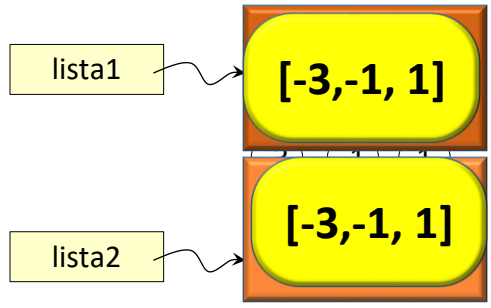


nome1, nome2 e nome3 tem o mesmo valor e se referem ao mesmo objeto

Copiar x Clonar (2/8)

lista1 = [-3,-1, 1]

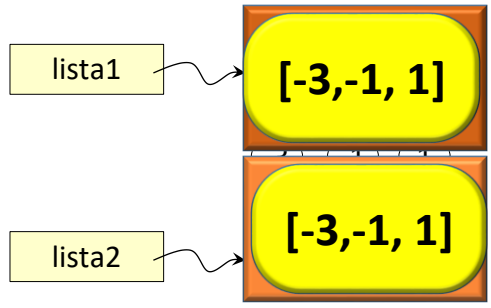
lista2 = [-3,-1, 1]



lista1 e lista2 tem o mesmo valor mas não se referem ao mesmo objeto

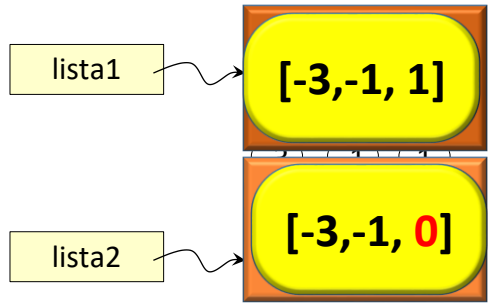
Copiar x Clonar (3/8)

lista2[2]=0



Copiar x Clonar (4/8)

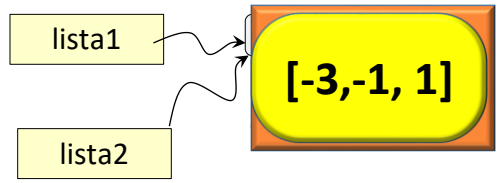
lista2[2]=0



Copiar x Clonar (5/8)

lista1 = [-3,-1, 1]

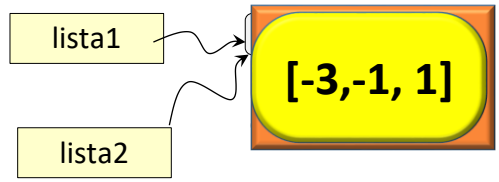
lista2 = lista1



A mesma lista tem dois nomes diferentes (apelidos da lista).
Qualquer mudança feita com um apelido afeta o outro.

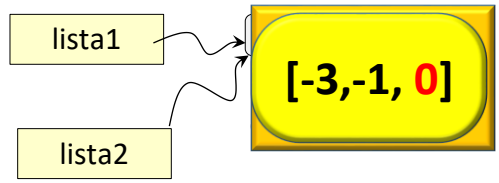
Copiar x Clonar (6/8)

lista2[2] = 0



Copiar x Clonar (7/8)

lista2[2] = 0

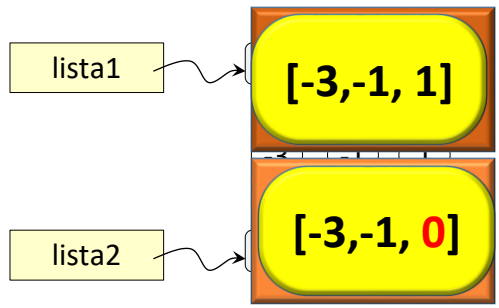


Qualquer mudança feita com um apelido afeta o outro.

Copiar x Clonar (8/8)

Para modificar uma lista e manter uma cópia da lista original deve-se **clonar.**
O operador de fatiação faz uma cópia→ clona

ista1 = [-3,-1, 1]
lista2 = lista1[:]



Voltando às sequências numéricas

```
pontos=[0,1,2,3,4,5,6,7,8,9,10]
valores = []
for x in pontos:
    valores.append(f(x))
copia = valores[ : ]
for i in range(2,10,3):
    copia[i]="_____"

adivinha(valores,copia)
```

Solução: sequências numéricas

```
def f(x):
   return x*100 + 67

def verificaValor(pseq,x):
    acertou=False
    while not acertou:
        num = int( input ("Digite o %d° valor: "%pseq))
        acertou = num == x
    return num

def adivinha(valores,copia):
   for i in range(3,10,3):
      print(copia)
      copia[i-1]=  verificaValor(i,valores[i-1]))
   return
```

Listas com elementos duplicados

Seja a lista

ldup = random.sample(range(-10,10),4)*4

Faça a função elimDuplicados, que recebe uma lista de inteiros e remove seus elementos duplicados.

Solução: elementos duplicados

```
def elimDuplicados(l):
    i=0
    while(i<len(l)):
        if(l[i] in l[i+1:]):
            l.pop(i)    # o i-ésimo elemento mudou!
        else:
            i=i+1
    print(l)
    return
```

Mais sobre listas

Função filter()

filter(function,list)

Filter utiliza uma função booleana (eg. a > 3) e filtra para uma nova lista somente os elementos que geraram *true*

```
def f(x):
      return x % 5 == 0 and x % 2 == 0

filter(f, range(2,25))

 [10,20]
```

Função map()

map(function,list)

Map itera sobre os elementos de "list" e aplica a cada um um deles "function", gerando uma nova lista com os resultados.

```
def duplica(x):
      return 2*x

map(duplica, range(2,6))

[4,6,8,10]
```

Função reduce()

reduce(function,list)

Reduce aplica a função nos 2 primeiros elementos da sequência. Na próxima iteração, aplica a função no terceiro elemento e no resultado anteriormente obtido, e assim sucessivamente.

```
def soma(x,y):
      return x+y

reduce(soma, range(6))     #Equivalente a sum(lista).
15
```

Fatiamento x Atribuição: Inclusão (1/7)

:

1. Atribuindo uma lista maior que a fatia:

   lista[a : b] = *lista2,*

   substitui os elementos da fatia [a,b) com os primeiros elementos da *lista2* e insere os demais.

2. Atribuindo uma lista à uma fatia vazia:

   fatia vazia

   lista[a : a] = *lista2,*

   introduz os elementos da *lista2* na posição *a* de *lista*

Fatiamento x Atribuição: Inclusão (2/7)

Exemplos: **print(lista)**

```
lista = ["a","d","k","h"]
lista[2:3] = ["f","g"]          ["a","d","f","g","h"]
lista[1:1] = ["b","c"]          ["a","b","c","d","f","g","h"]
lista[4:4] = ["e"]              ["a","b","c","d","e","f","g","h"]
```

fatia vazia

OBS: fatias definidas com incremento, não permitem inclusão. O tamanho da fatia deve ser igual à quantidade de elementos atribuídos, caso contrário o Python gera um erro

Fatiamento x Atribuição: Inclusão (3/7)

Qual a diferença?

```
l = [ ]
x = 1
y = 2
l[0:0] = [x,y]
l[0:0] = [ [x,y] ]
print (l)
```

Fatiamento x Atribuição: Inclusão (4/7)

Qual a diferença?

```
l = []
x =  1
y =  2
l[0:0] = [x,y]
l[0:0] = [ [x,y] ]
print (l)
```

Cria uma lista com 2 elementos: x e y

Fatiamento x Atribuição: Inclusão (5/7)

Qual a diferença?

```
l = []
x =  1
y =  2
l[0:0] = [x,y]
l[0:0] = [ [x,y] ]
print (l)
```

Insere os 2 elementos da lista no índice 0 de *l*

l ⟶ **[1,2]**

Fatiamento x Atribuição: Inclusão (6/7)

Qual a diferença?

```
l = []
x =  1
y =  2
l[0:0] = [x,y]
l[0:0] = [ [x,y] ]
print (l)
```

Cria uma lista com 1 elemento [x,y]

Fatiamento x Atribuição: Inclusão (7/7)

Qual a diferença?

```
l = []
x =  1
y =  2
l[0:0] = [x,y]
l[0:0] = [ [x,y] ]
print (l)
```

Insere o elemento da lista [[x,y]] no índice 0 da l

l ⟶ **[[1, 2] , 1 , 2]**

Fatiamento x Atribuição: Exclusão (1/4)

Lista[a : b] = []

Exclusão de elementos:

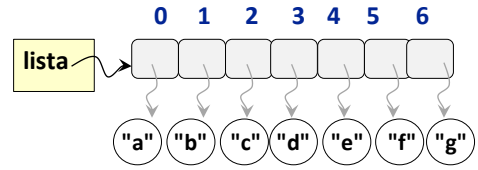
A atribuição da lista vazia à fatia, remove seus elementos

Exemplo 1:

lista = ["a", "b", "c", "d", "e", "f", "g"]

lista[4:5] = [ ]

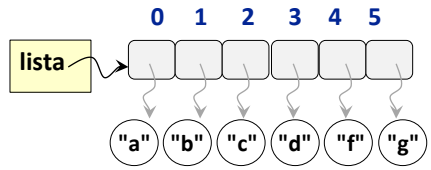
fatia com 5º elemento



Fatiamento x Atribuição: Exclusão (2/4)

Exemplo 1:

```
lista[4:5] = []          # retira o 5º elemento
```



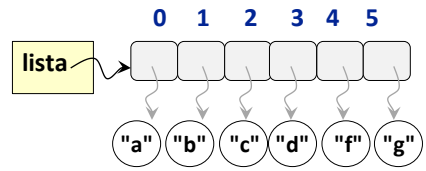

```
["a","b","c","d","f","g"]
```

Fatiamento x Atribuição: Exclusão (3/4)

Exemplo 2:

```
lista[1:3] = []
```

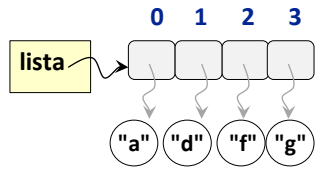
fatia com 2º e 3º elemento



Fatiamento x Atribuição: Exclusão (4/4)

Exemplo 2:

```
lista[1:3] = [ ]          # exclui 2º e 3º elementos
```




```
["a","d","f","g"]
```