MC-102 — Aula 10
Listas

Eduardo C. Xavier

Instituto de Computação – Unicamp

19 de Outubro de 2020

Roteiro

Listas

- Listas são construções de linguagens de programação que servem para armazenar vários dados de forma simplificada.
- No caso de Python, listas podem armazenar dados de um mesmo tipo (lista uniforme) ou dados de tipos diferentes.

Inicialização de Listas

- Em algumas situações é necessário declarar e já atribuir um conjunto de objetos para uma lista.
- Podemos usar o que é conhecido como **compreensão de listas**.
- Dentro da lista incluímos uma construção com um laço que gerará valores iniciais para a lista.

Inicialização de Listas

- A sintaxe básica é:

```
variavel = [expressão for i in range(n)]
```

Inicialização de Listas

- Exemplo: inicializar uma lista com 5 zeros:

```
>>> x = [ 0 for i in range(5)]
>>> x
[0, 0, 0, 0, 0]
```

- Exemplo: inicializar uma lista com os 5 primeiros números pares:

```
>>> x = [ 2*i for i in range(5)]
>>> x
[0, 2, 4, 6, 8]
```

Inicialização de Listas

- Exemplo: inicializar uma lista com as áreas de círculos de raio 3,6,9, e 12:

```
>>> import math
>>> l = [math.pi*r*r for r in range(3,13,3)]
>>> l
[28.27, 113.09, 254.46, 452.38]
```

Exemplo: Elementos Iguais

- Ler duas listas com 5 inteiros cada.
- Checar quais elementos da segunda lista são iguais a algum elemento da primeira lista.
- Se não houver elementos em comum, o programa deve informar isso.

Exemplo: Elementos Iguais

- Abaixo está o código que faz a leitura das listas.

```
v1 = [0 for i in range(5)]
v2 = [0 for i in range(5)]

#Lendo dados de v1
for i in range(5):
        v1[i] = int(input("v1[%d]: " %i))

print()

#Lendo dados de v2
for i in range(5):
        v2[i] = int(input("v2[%d]: " %i))

print(v1)
print(v2)
  ...
```

Exemplo: Elementos Iguais

- Para cada elemento de **v1** testamos todos os outros elementos de **v2** para saber se são iguais.
- Usamos uma variável indicadora para decidir ao final dos laços encaixados, se as listas possuem ou não um elemento em comum.

```
ele_comum = False #Assumimos que não hajam elementos comuns
for x1 in v1:
        for x2 in v2:
                if x1 == x2:
                        ele_comum = True #Descobrimos que há elemento comum
                        print("Elemento comum:", x1)

if not ele_comum:
        print("Listas não possuem elemento em comum!")
```

Exemplo: Elementos Iguais

- Código completo abaixo.

```
v1 = [0 for i in range(5)]
v2 = [0 for i in range(5)]

#Lendo dados de v1
for i in range(5):
        v1[i] = int(input("v1[%d]: " %i))

print()

#Lendo dados de v2
for i in range(5):
        v2[i] = int(input("v2[%d]: " %i))

print(v1)
print(v2)

ele_comum = False #Assumimos que não hajam elementos comuns
for x1 in v1:
        for x2 in v2:
                if x1 == x2:
                        ele_comum = True #Descobrimos que há elemento comum
                        print("Elemento comum:", x1)

if not ele_comum:
        print("Listas não possuem elemento em comum!")
```

Objetos Mutáveis e Imutáveis

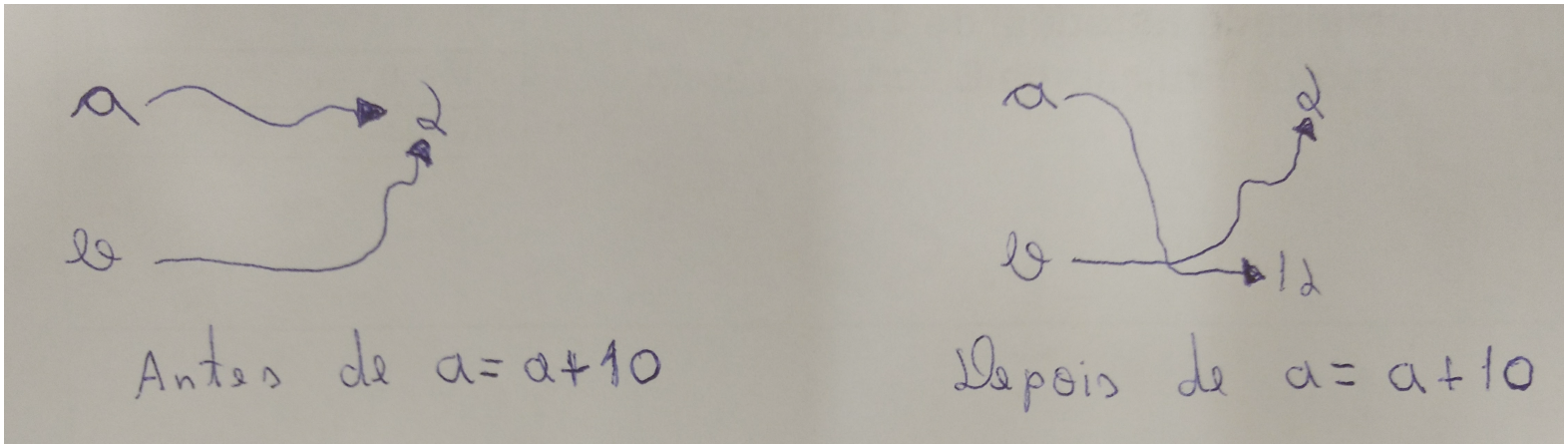
- Tudo em Python é um objeto, e seu tipo especifica que operações podem ser realizadas sobre o objeto.
- O tipo de um objeto também indica se ele é **mutável** ou **imutável**.
- Como o nome sugere, um objeto imutável não pode ser alterado.
  - Os tipos vistos anteriormente, **int, float, string, bool** são todos imutáveis.
- Já as listas são mutáveis, o que significa que operações podem ser realizadas sobre um objeto lista alterando-o (não é criado um novo objeto após a alteração).

Objetos Mutáveis e Imutáveis

- Considere o exemplo:

```
>>> a = 2
>>> b = a
>>> a = a + 10
>>> a
12
>>> b
2
```

- O tipo **int** é imutável, logo quando realizamos a soma **a+10** o objeto **2** não pode ser alterado, e o que ocorre é a criação de um novo objeto **12** e uma nova associação de **a** com este.

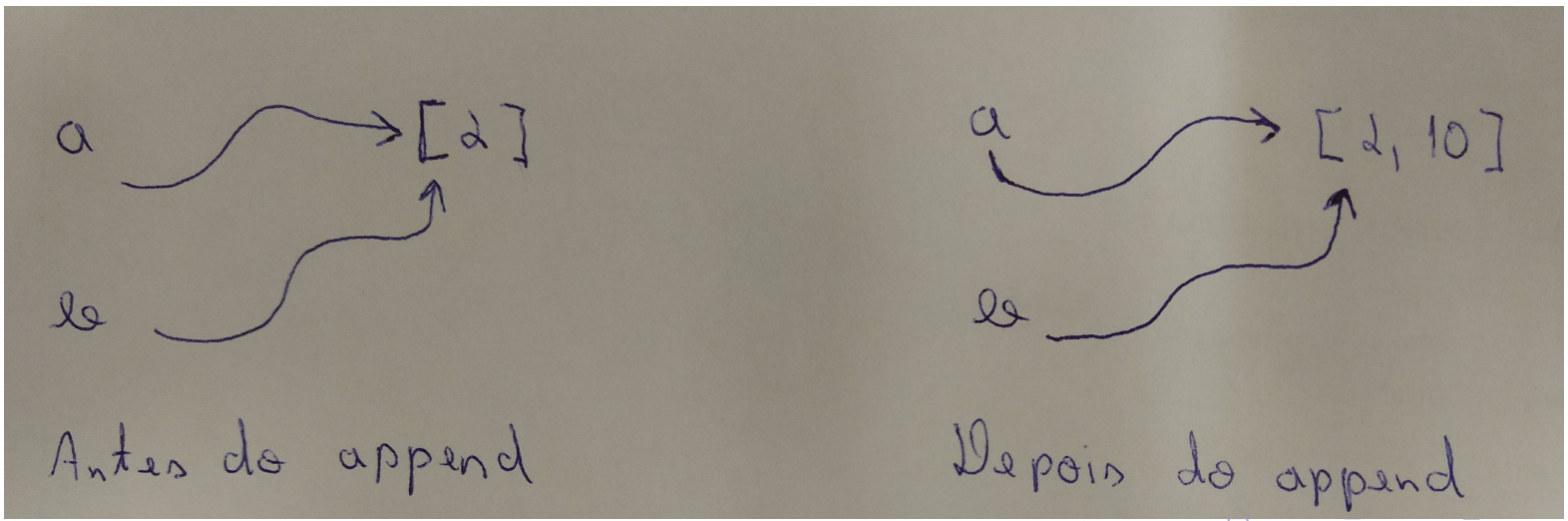


Objetos Mutáveis e Imutáveis

- Considere o exemplo com listas:

```
>>> a=[2]
>>> b=a
>>> a.append(10)
>>> a
[2, 10]
>>> b
[2, 10]
```

- O tipo **list** é mutável, logo quando executamos o método **a.append(10)** a lista é alterada e como **b** estava associada com a mesma lista, o resultado é percebido em **b** também:



Objetos Mutáveis e Imutáveis

- Alguns tipos são mutáveis pois a criação de objetos deste tipo é caro computacionalmente.
- A operação de concatenação (+) sobre listas cria sempre uma nova lista, o que é caro:

```
>>> a=[1,2,3,4]
>>> b=[5,6,7,8,9]
>>> c = a+b
>>> c
[1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> id(a)
4302774664
>>> id(b)
4302778312
>>> id(c)
4302774408
```

- A função **id** devolve o identificador do objeto que uma variável está associado.
- No exemplo **a**, **b**, e **c** estã associados com 3 objetos distintos.

Objetos Mutáveis e Imutáveis

- Se não é problema alterar uma das listas (**a** ou **b**) o método **extend** é preferível à concatenação:

```
>>> a=[1,2,3,4]
>>> b=[5,6,7,8,9]
>>> id(a)
4302774664
>>> a.extend(b)
>>> id(a)
4302774664
>>> a
[1, 2, 3, 4, 5, 6, 7, 8, 9]
```

- Note que **a** continuou associado com a mesma lista, e esta teve o conteúdo alterado.

Objetos Mutáveis e Imutáveis

- Por outro lado, em algumas situações não desejamos alterar um objeto que sofrerá alterações.
- Neste caso pode-se usar o método **copy** que cria uma copia do objeto especificado.

```
>>> a=[1,2,3,4]
>>> b = a.copy()
>>> id(a)
4302758856
>>> id(b)
4302758088
>>> a.append(10)
>>> a
[1, 2, 3, 4, 10]
>>> b
[1, 2, 3, 4]
```

Exercício

- Escreva um programa que lê 10 números inteiros e os salva em uma lista. Em seguida o programa deve encontrar a posição do maior elemento da lista e imprimir esta posição.

Exercício

- Escreva um programa que lê 10 números ponto flutuante e os salva em uma lista. Em seguida o programa deve calcular a média dos valores armazenados na lista e imprimir este valor.

Exercício

- Escreva um programa que lê 10 números inteiros e os salva em uma lista. Em seguida o programa deve ler um outro número inteiro C. O programa deve então encontrar dois números de posições distintas da lista cuja multiplicação seja C e imprimi-los. Caso não existam tais números, o programa deve informar isto.
- Exemplo: Se $l = (2, 4, 5, -10, 7)$ e $C = 35$ então o programa deve imprimir "5 e 7". Se $C = -1$ então o programa deve imprimir "Não existem tais números".

Exercício

- Escreva um programa que lê duas listas de 5 inteiros e salva em uma terceira lista os elementos de qualquer uma das duas listas que não está na outra lista (a ordem não é importante).
- Exemplo: Se $l_1 = [2, 4, 3]$ e $l_2 = [1, 2, -10, 4, 5]$ então a a terceira lista deve conter $l_3 = [3, 1, -10, 5]$ ou uma lista com os mesmos elementos em outra ordem.