Computação I - Python
Aula 6: Fatiamento e Manipulação de Listas
Remoção de Elementos de Listas

Apresentado por: Rafael Machado Andrade

Produção DCC-UFRJ

Metodologia de referência https://doi.org/10.5753/wei.2016.9683

Remoção de elementos de listas

- **Comando del**: Remove elementos de uma lista a partir de seu índice.
  Utilização: **del lista[índice]** ou **del lista[fatiamento]**

Remoção de elementos de listas

- **Comando del**: Remove elementos de uma lista a partir de seu índice.
  Utilização: **del lista[índice]** ou **del lista[fatiamento]**

```
>>> L = [1,2,3,4,5]
>>> del L[0]
>>> L
[2, 3, 4, 5]
```

O elemento na posição 0 da lista será removido (ou seja, 1).

Ao se especificar um fatiamento ao invés de um único índice, todos os elementos que fazem parte dessa fatia serão removidos da lista:

```
>>> L = [1,2,3,4,5]
>>> del L[::2]
>>> L
[2, 4]
```

Os elementos da fatia [::2] serão removidos da lista (ou seja, 1, 3 e 5).

Remoção de elementos de listas

OBS: Após removermos um ou mais elementos de uma lista, essa lista ficará com um tamanho menor do que anteriormente, e com isso todos os elementos existentes após os elementos removidos terão seus índices reduzidos.

```
>>> L = [1, 2, 3, 4, 5]
>>> len(L)      #retorna o tamanho da lista
5
>>> L[3]        #obtendo o elemento na posicao 3 da lista
4
>>> del L[2]    #removendo o elemento na posicao 2 da lista
>>> L
[1, 2, 4, 5]
>>> len(L)      #retorna o tamanho da lista
4
>>> L[3]        #obtendo o elemento na posicao 3 da lista
5
```

A lista original possuía tamanho 5 e o elemento na posição 3 era o número 4.
Após remover o elemento na posição 2, o tamanho da lista se tornou 4,
e o elemento na posição 3 agora é o número 5.

Remoção de elementos de listas

OBS: É possível remover elementos de uma lista contida em outra lista ao se utilizar colchetes em sequência, referenciando os índices desejados:

```
>>> M = [[1, 2, 3], ['a', 'b', 'c'], '1a2b3c']
>>> del M[0][-1]
>>> M
[[1, 2], ['a', 'b', 'c'], '1a2b3c']
>>> del M[1][1]
>>> M
[[1, 2], ['a', 'c'], '1a2b3c']
```

Tentar fazer o mesmo com um elemento do tipo str causará um erro de tipo:

```
>>> M = [[1, 2, 3], ['a', 'b', 'c'], '1a2b3c']
>>> del M[2][0]
Traceback (most recent call last):
  File "<pyshell#46>", line 1, in <module>
    del M[2][0]
TypeError: 'str' object doesn't support item deletion
```

Remoção de elementos de listas

OBS: Caso seja especificado um índice inexistente, ocorrerá um erro de índice:

```
>>> L = [1, 2, 3, 4, 5]
>>> del L[10]
Traceback (most recent call last):
  File "<pyshell#48>", line 1, in <module>
    del L[10]
IndexError: list assignment index out of range
```

Remoção de elementos de listas

OBS: Caso seja especificado um índice inexistente, ocorrerá um erro de índice:

```
>>> L = [1, 2, 3, 4, 5]
>>> del L[10]
Traceback (most recent call last):
  File "<pyshell#48>", line 1, in <module>
    del L[10]
IndexError: list assignment index out of range
```

Uma outra forma de se remover um elemento de uma lista através de sua posição é utilizando-se uma função das listas, a função **POP**.

Remoção de elementos de listas

- **Função POP**: Remove um elemento de uma lista, de um índice específico. Utilização: **list.pop(lista, índice)**

Remoção de elementos de listas

- **Função POP**: Remove um elemento de uma lista, de um índice específico. Utilização: **list.pop(lista, índice)**

```
>>> L = [1, 2, 3, 4, 5]
>>> list.pop(L, 0)
1                              #retorno da funcao pop
>>> L
[2, 3, 4, 5]
```

O elemento na posição 0 da lista será removido e retornado (ou seja, 1).

Remoção de elementos de listas

Observe que a função **pop** retorna o valor removido da lista, diferentemente do comando **del**, que não gera retorno algum. Dessa forma, podemos "armazenar" o valor removido ao atribuí-lo a uma variável.

```
>>> L = [1, 2, 3, 4, 5]
>>> removido = list.pop(L, 2)
>>> L
[1, 2, 4, 5]
>>> removido
3
```

O elemento na posição 2 da lista será removido e salvo na variável 'removido'.

Remoção de elementos de listas

OBS: Caso não seja especificado um índice para a função **pop**, será assumido o índice -1, ou seja, será removido o último elemento da lista:

```
>>> L = [1, 2, 3, 4, 5]
>>> list.pop(L)
5                              #retorno da funcao pop
>>> L
[1, 2, 3, 4]
```

O elemento na última posição da lista será removido e retornado (ou seja, 5).

Remoção de elementos de listas

OBS: Assim como ocorria com o comando **del**, caso seja especificado um índice inexistente para a função **pop**, ocorrerá um erro de índice:

```
>>> L = [1, 2, 3, 4, 5]
>>> list.pop(L, 10)
Traceback (most recent call last):
  File "<pyshell#57>", line 1, in <module>
    list.pop(L, 10)
IndexError: pop index out of range
```

Remoção de elementos de listas

OBS: Assim como ocorria com o comando **del**, caso seja especificado um índice inexistente para a função **pop**, ocorrerá um erro de índice:

```
>>> L = [1, 2, 3, 4, 5]
>>> list.pop(L, 10)
Traceback (most recent call last):
  File "<pyshell#57>", line 1, in <module>
    list.pop(L, 10)
IndexError: pop index out of range
```

Podemos também remover um elemento de uma lista sem especificar uma posição, e sim especificando qual é o elemento a ser removido. Para isso, podemos utilizar outra função das listas, a função **REMOVE**.

Remoção de elementos de listas

- **Função REMOVE**: Remove a primeira ocorrência de um elemento de uma lista. Utilização: **list.remove(lista, elemento)**

Remoção de elementos de listas

- **Função REMOVE**: Remove a primeira ocorrência de um elemento de uma lista. Utilização: **list.remove(lista, elemento)**

```
>>> R = [1, 2, 3, 4, 3, 2, 1, 1]
>>> list.remove(R, 2)
>>> R
[1, 3, 4, 3, 2, 1, 1]
```

A primeira ocorrência do elemento 2 será removida da lista (ou seja, da posição 1).

Observe que há mais ocorrências do elemento 2 na lista, porém somente a primeira é removida. Caso deseje-se remover a ocorrência seguinte, será necessário executar o comando **remove** novamente.

Remoção de elementos de listas

OBS: Caso o elemento especificado não exista na lista, ocorrerá um erro de valor:

```
>>> R = [1, 2, 3, 4, 3, 2, 1, 1]
>>> list.remove(R, 8)
Traceback (most recent call last):
  File "<pyshell#66>", line 1, in <module>
    list.remove(R, 8)
ValueError: list.remove(x): x not in list
```

Remoção de elementos de listas

OBS: Caso o elemento especificado não exista na lista, ocorrerá um erro de valor:

```
>>> R = [1, 2, 3, 4, 3, 2, 1, 1]
>>> list.remove(R, 8)
Traceback (most recent call last):
  File "<pyshell#66>", line 1, in <module>
    list.remove(R, 8)
ValueError: list.remove(x): x not in list
```

Podemos utilizar o operador **in** para verificar se um elemento existe em uma lista, antes de tentar removê-lo com a função **remove**.

- **elemento in lista** : retorna True se 'elemento' existe na 'lista',
  retorna False caso contrário

Remoção de elementos de Listas

Exemplo: Faça uma função que receba uma lista e um número como argumentos, remova a primeira ocorrência desse número da lista apenas caso ele exista na lista recebida e por fim retorne a lista resultante.

Remoção de elementos de Listas

Exemplo: Faça uma função que receba uma lista e um número como argumentos, remova a primeira ocorrência desse número da lista apenas caso ele exista na lista recebida e por fim retorne a lista resultante.

```
def removeNumero(lista, num):
    """Funcao que remove um numero de uma lista
    desde que esse numero exista na lista.
    list, float -> list """
    if num in lista:
        list.remove(lista, num)
    return lista
```

Testando a função com 2 valores: um existente, e um não existente na lista:

```
>>> R = [1, 2, 3, 4, 3, 2, 1, 1]
>>> removeNumero(R, 3)
[1, 2, 4, 3, 2, 1, 1]
>>> removeNumero(R, 8)
[1, 2, 4, 3, 2, 1, 1]
```

Remoção de elementos de listas

- **Função CLEAR**: Remove todos os elementos de uma lista.
  Utilização: **list.clear(lista)**

Remoção de elementos de listas

- **Função CLEAR**: Remove todos os elementos de uma lista.
  Utilização: **list.clear(lista)**

```
>>> M = [[1, 2, 3], ['a', 'b', 'c'], 'a1b2c3']
>>> list.clear(M)
>>> M
[]
```

Todos os elementos da lista são removidos de uma só vez com a função clear.

Remoção de elementos de listas

RECAPITULANDO...

- **Comando del**: Remove elementos de uma lista a partir de seu índice.
  Utilização: **del lista[índice]** ou **del lista[fatiamento]**
- **Função POP**: Remove um elemento de uma lista, de um índice específico.
  Utilização: **list.pop(lista, índice)**
- **Função REMOVE**: Remove a primeira ocorrência de um elemento de uma lista.
  Utilização: **list.remove(lista, elemento)**
- **Função CLEAR**: Remove todos os elementos de uma lista.
  Utilização: **list.clear(lista)**

Autores

- João C. P. da Silva ▸ Lattes
- Carla Delgado ▸ Lattes
- Ana Luisa Duboc ▸ Lattes

Colaboradores

- Anamaria Martins Moreira ▸ Lattes
- Fabio Mascarenhas ▸ Lattes
- Leonardo de Oliveira Carvalho ▸ Lattes
- Charles Figueiredo de Barros ▸ Lattes
- Fabrício Firmino de Faria ▸ Lattes

Computação I - Python
Aula 6: Fatiamento e Manipulação de Listas
Remoção de Elementos de Listas

Apresentado por: Rafael Machado Andrade

Produção DCC-UFRJ

Metodologia de referência https://doi.org/10.5753/wei.2016.9683