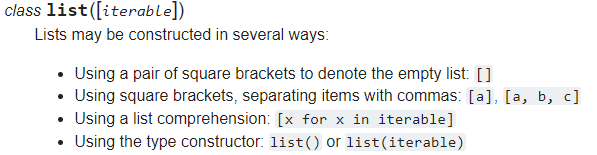
38 – Listas

Lista é um tipo de mutable sequence que pode armazenar objetos em sequência, para uma definição mais apropriada veja:

<https://docs.python.org/3/library/stdtypes.html#typesseq> (É super recomendado ler esta documentação e varias coisas que falamos aqui está lá)

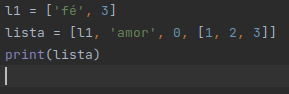


Ela pode armazenar em sequência diversos tipos de dados:

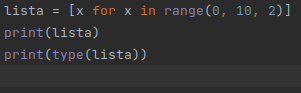


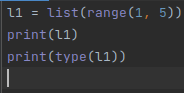


Listas podem armazenar até outras listas e variáveis:

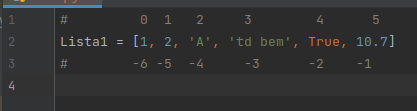
 

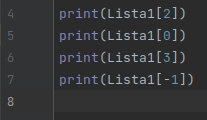
Outras formas de construir a lista:

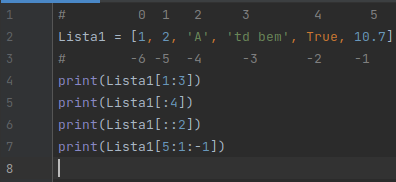
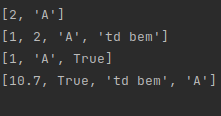
 

Os seus itens são guardados em sequência e endereçados por meio de índices (como em string)

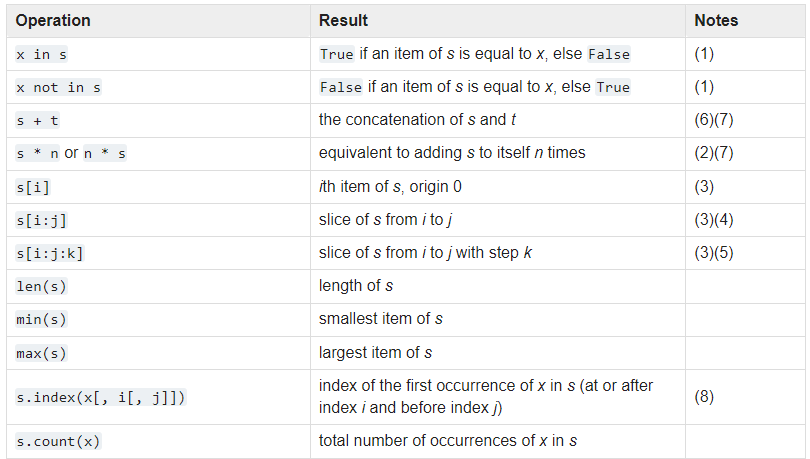


Podemos fazer também o fatiamento da lista (como em string):

**Commun Operations com Listas**

****

Notes:

8.

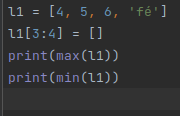
index raises [ValueError](https://docs.python.org/3/library/exceptions.html#ValueError) when x is not found in s. Not all implementations support passing the additional arguments i and j. These arguments allow efficient searching of subsections of the sequence. Passing the extra arguments is roughly equivalent to using s[i:j].index(x), only without copying any data and with the returned index being relative to the start of the sequence rather than the start of the slice.

\*1.

A sintaxe do index é:

s.index(x, i, j), sendo o i e o j opcionais (default i=, j=)

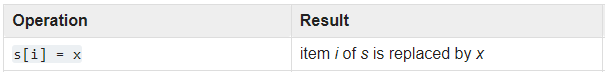
Examples:

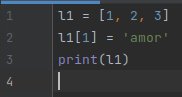
 

**Muttable Operations com Listas e Funções que modificam listas**

<https://docs.python.org/3/library/stdtypes.html#immutable-sequence-types>

**Atribuição de um item da lista**



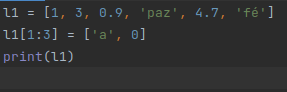
 

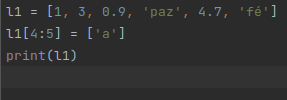
**Atribuição de um pedaço da lista**



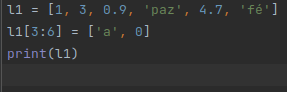
O pedaço da lista de s[i] até s[j-1] é substituido pela lista t.

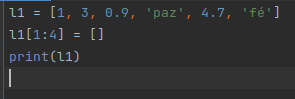
Caso o pedaço s[i]-s[j-1] tenha o mesmo tamanho de t

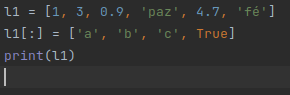
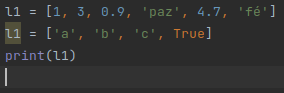
 

Caso o pedaço s[i]-s[j-1] for maior que a lista t (todo o pedaço s[i]-s[j-1] será substituido pela lista t, tornando a lista s resultante menor do que era)

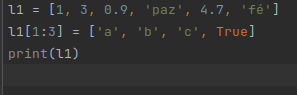
 

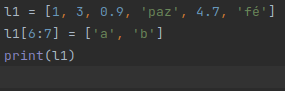
 ou 



Caso j>i

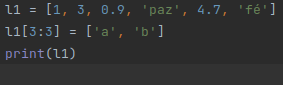
Caso o pedaço s[i]-s[j-1] for menor que a lista t[0]-t[k-1] (todo o pedaço s[i]-s[j-1] será substituido por t[0]-t[j-i-1], os itens s[j]-s[n-1] serão empurrados pra direita ficando depois do pedaço t[j-i]-t[k-1] que ficarão nas posições s[j]-s[k+i-1])

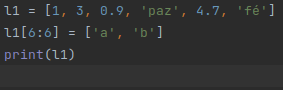


Caso j==i

O conteúdo s[i]-s[n-1] será empurrado para direita ficando depois do pedaço t[0]-t[k-1] que ficará na posição s[i]-[i+k-1]. (Note que portanto nenhum item da lista original s é removido)

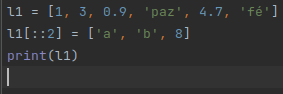
 

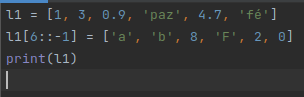


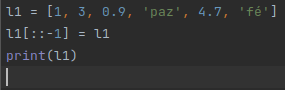


Podemos acrescentar o step

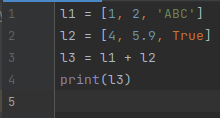


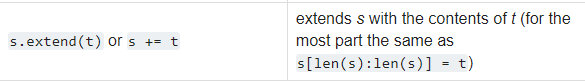
 

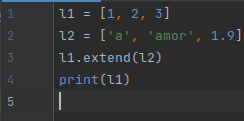
**Podemos concatenar listas:**

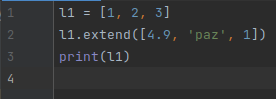
 

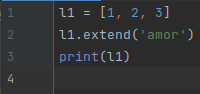
**<lista>.extend(<iterable t>)**



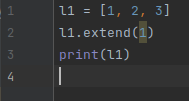
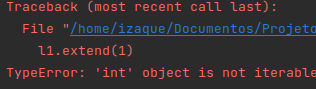
Extende a lista concatenando com o <iterable t> que pode ser uma string ou uma lista por exemplo.

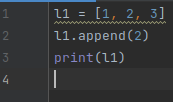
Repare que cada item da string é um caracter, e ele adiciona cada item da string como um item da lista.

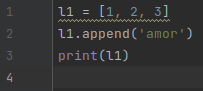
 

**<lista>.append(x)**

Acrescenta o valor (dado primitivo int, float, bool, string) na lista.

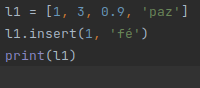


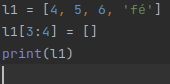
**<lista>.insert(i, x)**

****



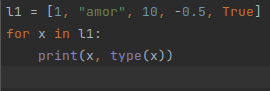
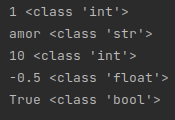
**del <lista>[i:j]**



**Adicionais:**

Como já sabemos uma sequência é um objeto iterável, logo podemos utilizar uma lista no for:

**Listas como Matrizes:**

