https://pythonhelp.wordpress.com/2013/06/26/brincando-com-listas/

**BRINCANDO COM LISTAS**

Criando uma lista de números em sequência:

# python 2:

lista = range(100)

# python 3:

lista = list(range(100))

Criando uma lista com [*list comprehensions*](https://pythonhelp.wordpress.com/2011/03/01/list-comprehension/):

lista = [x\*2 for x in range(100)]

Percorrendo uma lista com for in:

for x in lista:

# faça algo com x

Percorrendo uma lista, [obtendo os valores e seus índices](https://pythonhelp.wordpress.com/2012/01/05/a-funcao-enumerate/):

for indice, valor in enumerate(lista):

print "lista[%d] = %d" % (indice, valor)

Percorrendo um pedaço de uma lista usando [*slicing*](https://pythonhelp.wordpress.com/2011/11/12/fatiamento-slicing-de-strings-em-python/):

for x in lista[40:60]:

# faça algo com x

Percorrendo uma lista de trás pra frente definindo o passo do *slicing* como -1:

for x in lista[::-1]:

# faça algo com x

Ou:

for x in reversed(lista):

# faça algo com x

Percorrendo uma lista ordenada:

for x in sorted(lista):

# faça algo com x

Acessando o [último elemento de uma lista](https://pythonhelp.wordpress.com/2011/12/09/acessando-o-ultimo-elemento-de-uma-lista/) com o índice -1:

print lista[-1]

Copiando uma [**referência**](https://pythonhelp.wordpress.com/2013/02/20/variaveis-valores-e-referencias/) para uma lista:

>>> nova\_ref = lista

>>> nova\_ref is lista

True

Copiando *de verdade* uma lista:

>>> nova\_lista = lista[:]

>>> nova\_lista is lista

False

Ou, usando o módulo copy:

>>> import copy

>>> nova\_lista = copy.copy(lista)

>>> nova\_lista is lista

False

Ou caso lista contivesse listas aninhadas e quiséssemos fazer com que estas também fossem completamente copiadas, e não somente referenciadas, usaríamos a função [deepcopy()](http://docs.python.org/2/library/copy.html#copy.deepcopy):

>>> nova\_lista = copy.deepcopy(lista)

>>> nova\_lista is lista

False

Embaralhando os elementos de uma lista (*in-place*):

>>> import random

>>> random.shuffle(lista) # altera a própria lista

Obtendo uma amostra *aleatória* (de 10 elementos) de uma lista:

>>> print random.sample(lista, 10)

[729, 9025, 2401, 8100, 5776, 784, 1444, 484, 6241, 7396]

Obtendo um elemento *aleatório* de uma lista:

>>> random.choice(lista)

7744

Gerando uma lista com 10 números *aleatórios*, com valores entre 0 e 99:

>>> lista\_aleatoria = random.sample(range(0, 100), 10)

Obtendo o maior elemento de uma lista:

>>> lista = range(0, 10)

>>> print max(lista)

9

O menor:

>>> print min(lista)

0

Pegando somente os elementos de **índice par**:

>>> print lista[::2]

[0, 2, 4, 6, 8]

Os de **índice ímpar**:

>>> print lista[1::2]

[1, 3, 5, 7, 9]

Somando todos os elementos de uma lista:

>>> print sum([1, 2, 3, 4])

10

[Juntando duas listas](https://pythonhelp.wordpress.com/2013/04/16/funcao-zip-em-python/), formando pares de elementos:

>>> lista = zip(range(0, 5), range(5, 10))

>>> print lista

[(0, 5), (1, 6), (2, 7), (3, 8), (4, 9)]

Separando os elementos de uma lista de forma intercalada:

>>> lista = range(0, 10)

>>> intercaladas = lista[::2], lista[1::2]

>>> print intercaladas

([0, 2, 4, 6, 8], [1, 3, 5, 7, 9])

Transformando uma lista de strings em uma string [CSV](https://pt.wikipedia.org/wiki/Comma-separated_values):

>>> lista = ["ola", "mundo", "aqui", "estamos"]

>>> csv\_values = ','.join(lista)

>>> print csv\_values

ola,mundo,aqui,estamos

[Aplicando uma função (neste caso, **anônima**)](https://pythonhelp.wordpress.com/2012/05/13/map-reduce-filter-e-lambda/) a todos elementos de uma lista:

>>> lista = range(1, 11)

>>> print map(lambda x: x\*-1, lista)

[-1, -2, -3, -4, -5, -6, -7, -8, -9, -10]

Filtrando os elementos de uma lista de acordo com um critério:

>>> def criterio(x): return x >= 0

>>> print range(-5, 5)

[-5, -4, -3, -2, -1, 0, 1, 2, 3, 4]

>>> print filter(criterio, range(-5, 5))

[0, 1, 2, 3, 4]

Retirando os elementos cujo valor é *zero* ([ou melhor](https://pythonhelp.wordpress.com/2013/05/23/retirando-elementos-vazios-de-uma-lista/), cujo valor é avaliado como False):

>>> print filter(None, range(-2, 2))

[-2, -1, 1]