

In [24]: *# 1 - Crie uma Lista com elementos e calcule a terceira potencia de cada elemento.*

```
lista = [3,4,5,10,50,15]
pot = [ item**3 for item in lista ]
print (pot)
```

[27, 64, 125, 1000, 125000, 3375]

In [31]: *# 2 - Reescreva o codibo usando a função map(). O resultado deve ser o mesmo.*

```
palavra = 'Programa de estágio 2022 na Cummins Brasil'.split()
resultado = [[w.upper( ), w.lower( ), len(w)] for w in palavra]
for i in resultado :
    print(i)
```

```
['PROGRAMA', 'programa', 8]
['DE', 'de', 2]
['ESTÁGIO', 'estágio', 7]
['2022', '2022', 4]
['NA', 'na', 2]
['CUMMINS', 'cummins', 7]
['BRASIL', 'brasil', 6]
```

In [35]: *# resultado:*
 resultado = map(lambda w:[w.upper(),w.lower(),len(w)], palavra)
 for i in resultado:
 print(i)

```
['PROGRAMA', 'programa', 8]
['DE', 'de', 2]
['ESTÁGIO', 'estágio', 7]
['2022', '2022', 4]
['NA', 'na', 2]
['CUMMINS', 'cummins', 7]
['BRASIL', 'brasil', 6]
```

In [9]: *# 3 - Calcule a matriz trnasposta da matriz abaixo:*
 matrix = [[1,2],[3.4],[5,6],[7,8]]
 transpose = [[row[i] for row in matrix] for i in range(2)]:
 print (transpose)

```
File "<ipython-input-9-002742bb83ea>", line 3
    transpose = [[row[i] for row in matrix] for i in range(2)]:
                  ^
```

SyntaxError: invalid syntax

In [36]: *# 4 - Crie duas funções, uma para elevar um número ao quadrado e outra para elevar ao cubo:
 # aplicação simultanea aos elementos da lista:*

```
lista = [2,4,8,13]

def square (lst):
    return (lst**2)

def cube(lst):
    return (lst**3)
funcs = [square, cube]
for i in lista:
    valor = map(lambda lst: lst(i), funcs)
    print (list (valor))
```

```
[4, 8]
[16, 64]
[64, 512]
[169, 2197]
```

In [10]: *# 5 - Conforme as Listas abaixo, faça com que cada elemento de A seja elevado ao elemento correspondente de B:*

```
a = [ 2,3,4,5]
b = [ 2,5,10,15]

list(map(pow, a, b))
```

Out[10]: [4, 243, 1048576, 30517578125]

In [14]: *# 6 - Considere o RANGE de valores abaixo e use a função FILTER() para retornar apenas os valores negativos:*

```
range(-20, 20)
list(filter((lambda x: x < 0), range (-20,20)))
```

Out[14]:

```
[-20,
-19,
-18,
-17,
-16,
-15,
-14,
-13,
-12,
-11,
-10,
-9,
-8,
-7,
-6,
-5,
-4,
-3,
-2,
-1]
```

In [5]: *# 7 - Usando a função FILTER() encontre os valores que são comuns as duas listas:*

```
a = [1,2,3,4,5,6,7,8,9,10,11,12,14,15,16,20]
b = [2,3,5,6,8,4,5,8,10,11,21,22,23,3,4,5,8]
```

```
print(list(filter(lambda x: x in a, b)))
```

```
[2, 3, 5, 6, 8, 4, 5, 8, 10, 11, 3, 4, 5, 8]
```

In [10]: *# 8 - considere o código e obtenha o mesmo valor usando o pacote time:*

```
import datetime
print(datetime.datetime.now().strftime('%d/%dm/%Y %H:%M'))
```

```
07/07m/2022 09:47
```

In [11]: *# resposta :*

```
import time
print(time.strftime( '%d/%dm/%Y %H:%M' ))
```

```
07/07m/2022 09:48
```

In [27]: *# 9 - Considerando os DICIONÁRIOS abaixo crie um terceiro com as chaves do dic 1 e valores do dic 2:*

```
a = {'a':1,'b':2,'c':3}
b = {'d':4,'e':5,'f':6}
```

```
def inverterPosicoes(c1,c2):
    dicTemp[c1key]= c2val

    for aKey, bval in zip(c1,c2.values()):
        dicTemp [aKey] = bval

    return dicTemp
```

```
dict3 = inverterPosicoes (a,b)
print(dict3)
```

```
-----
NameError                                Traceback (most recent call last)
```

```
<ipython-input-27-94e58601373c> in <module>
```

```
11     return dicTemp
```

```
12
```

```
----> 13 dict3 = inverterPosicoes (a,b)
```

```
14 print(dict3)
```

```
<ipython-input-27-94e58601373c> in inverterPosicoes(c1, c2)
```

```
4
```

```
5 def inverterPosicoes(c1,c2):
```

```
----> 6     dicTemp[c1key]= c2val
```

```
7
```

```
8     for aKey, bval in zip(c1,c2.values()):
```

```
NameError: name 'c2val' is not defined
```

In [38]: *# 10 - Considere a Lista abaixo e retorne apenas os elementos com indices maior que 5:*

```
lista = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i']

for indice, valor in enumerate(lista):
    if indice <= 5:
        continue
    else:
        print(valor, ' ', end='')
```

g h i

In []: