



escola  
britânica de  
artes criativas  
& tecnologia

# Python para análise de dados



# PROGRAMAÇÃO FUNCIONAL



## GUIA DA AULA 4



# Estude a função reduce

- Definição
- Funções de alta ordem
- Compossibilidade



Acompanhe aqui  
os temas que  
serão tratados  
na videoaula



## 1. Definição

Aplica uma função a todos os elemento de uma coleção, dois a dois, e retorna **apenas** um elemento.

```
variavel = reduce(função, coleção)
```

```
In [ ]: numeros = [1, 2, 3]
```

```
In [ ]: from functools import reduce

soma = reduce(lambda x, y: x + y, numeros)
print(soma)
```



## 2. Funções de alta ordem

**Exemplo:** Encontrar maior número em uma lista.

```

In [ ]: def maior_entre(primeiro: int, segundo: int) -> int:
        return primeiro if primeiro >= segundo else segundo

primeiro = 11
segundo = 11

print(maior_entre(primeiro=primeiro, segundo=segundo))
  
```

```

In [ ]: from random import random

print(random())
  
```



## 2. Funções de alta ordem

```

In [ ]: from random import random

numeros = [round(100 * random()) for _ in range(0,
100)] print(numeros)
  
```

```

In [ ]: maior_numero = reduce(maior_entre, numeros)
print(maior_numero)
  
```

```

In [ ]: maior_numero = reduce(
    lambda primeiro, segundo: primeiro if primeiro >= segundo else
    segundo, numeros
)
print(maior_numero)
  
```



### 3. Compossibilidade

**Exemplo:** Combinação de métodos funcionais.

```

In [ ]: from random import random

numeros = [round(100 * random()) for _ in range(0,
100)] print(numeros)
  
```

Eleve os números ao quadrado:

```

In [ ]: numeros_ao_quadrado = map(lambda numero: numero ** 2, numeros)
  
```



### 3. Compossibilidade

Filtra os números ímpares:

```
In [ ]: numeros_impares = filter(lambda numero: numero % 2 != 0, numeros_ao_quadrado)
```

Soma todos os números:

```
In [ ]: soma_numeros = reduce(lambda x, y: x + y, numeros_impares)
print(soma_numeros)
```





### 3. Compossibilidade

Todos os métodos de uma vez:

In [ ]:

```

soma_numeros = reduce(lambda x, y: x + y,
                      filter(lambda numero: numero % 2 !=
                              0, map(lambda numero: numero **
                                      2,
                                      numeros
                                      )
                              )
                      )

print(soma_numeros)

```

