Por que não falar de programação funcional?

Eduardo Mendes

Nome:
Eduardo Mendes

Email:
mendesxeduardo@gmail.com

GIT:
github.com/z4r4tu5tr4

Por onde começamos?

- Functors, monads, monoids, catamorphism
- Relações matemáticas
- Teoria das categorias

Por onde começamos?

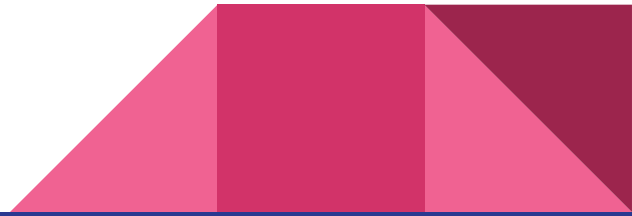
- Fu mon noids, cata
- Rela icas
- To cat

Uma leve introdução (sem imports)

- Funções, funções e funções
- Objetos de primeira classe
- Mapeamento
- Redução
- Lazy
- Closures

Primeiro, qual o objetivo dessa talk?

Estou montando um material, pretendo gravar uns vídeos sobre o assunto e queria saber o interesse da galera em aprender programação funcional



Em que isso pode mudar minha vida?

- Existem coisas que sei que sei
- Coisas que sei que não sei
- Coisas que não sei que não sei

Onde quero chegar com isso? [0]

Imagine fazer um tratado sobre cavalos (para quem nunca viu tal animal) que não comece pelo conceito de “cavalo” e sim pelo de “automóvel” (...). No fim cavalos serrão apenas aquilo que não são.

Onde quero chegar com isso? [1]

Aumento da capacidade de expressar ideias:

“É difícil para pessoas conceber estruturas que não podem descrever, verbalmente ou por escrito”

Onde quero chegar com isso? [2]

Imagine uma função que concatena dois iteráveis do mesmo tamanho usando um operador matemático devolvendo um novo iterável preguiçoso com o processamento gerado pela função.

Qual seria o nome dessa função?

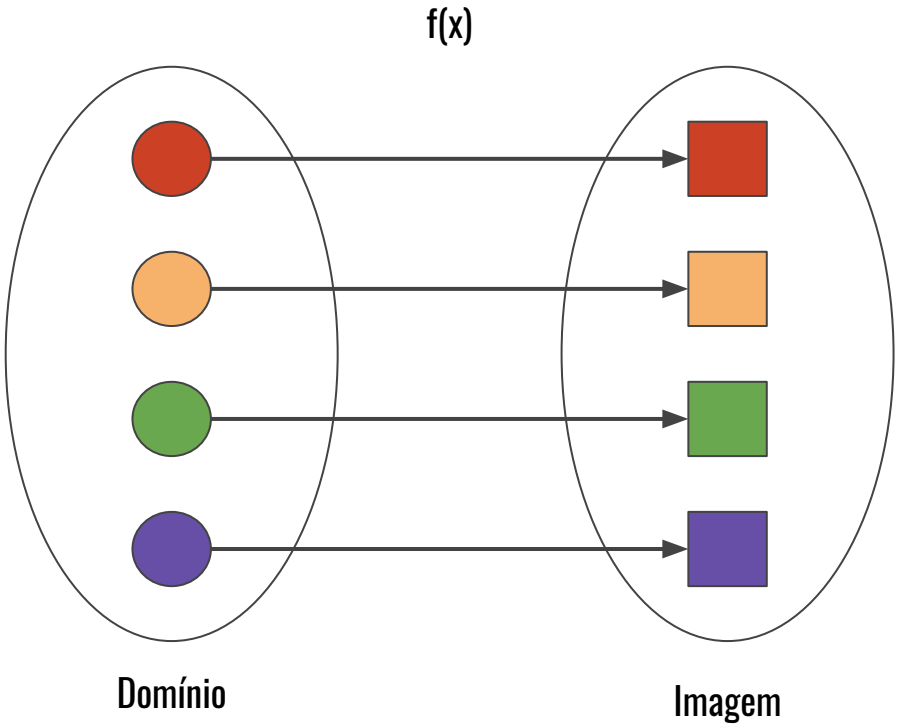
Onde quero chegar com isso? [3]

De toda maneira que eu possa explicar, a resposta nunca vai ser isso:

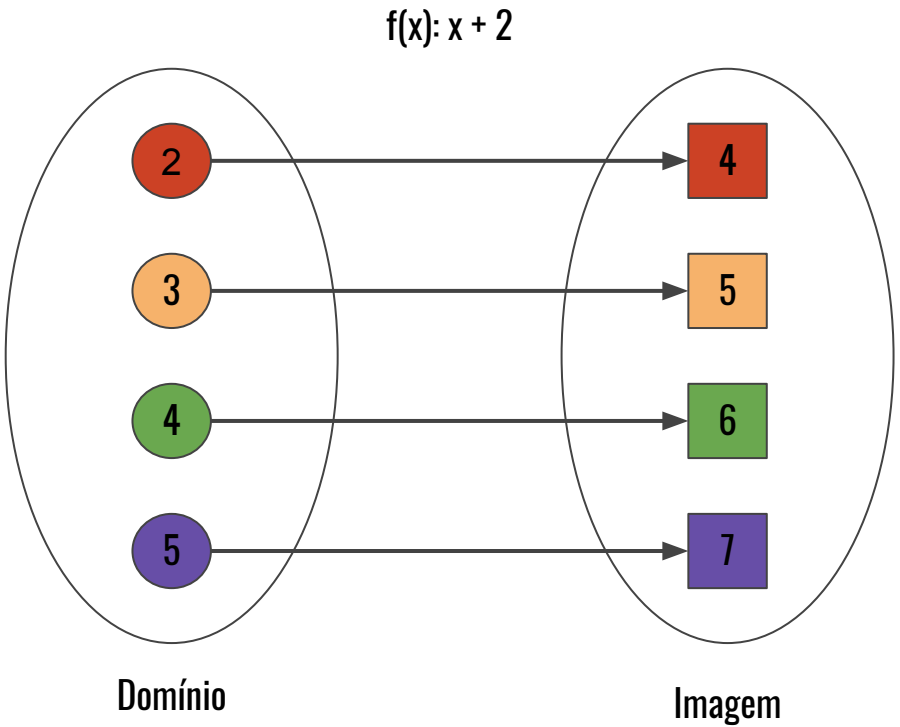
```
>>> list(zipwith(+, [1, 2, 3, 4], [1, 2, 3, 4]))
# [2, 4, 6, 8]
```

Então, antes de tudo. Vamos falar de funções e sequências

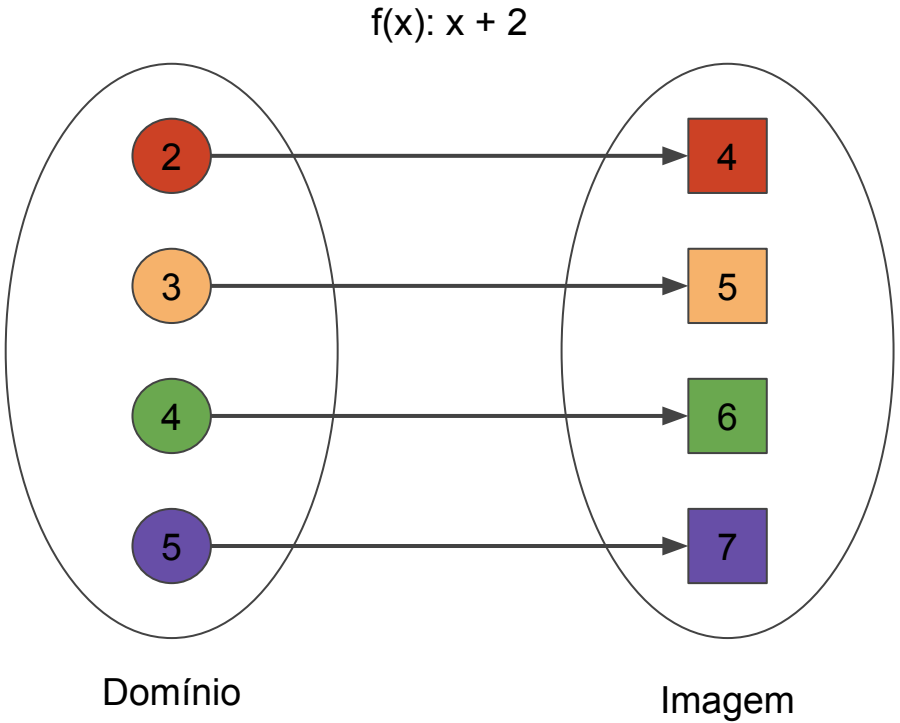
Funções, funções e funções



Funções, funções e funções

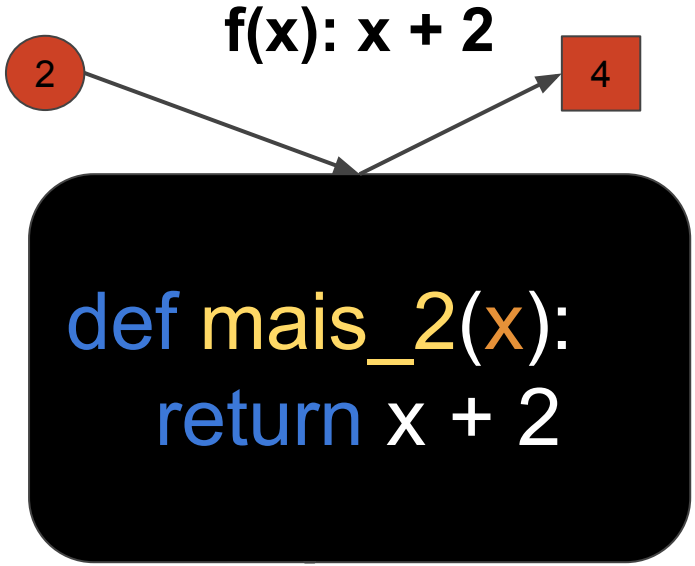


Funções, funções e funções



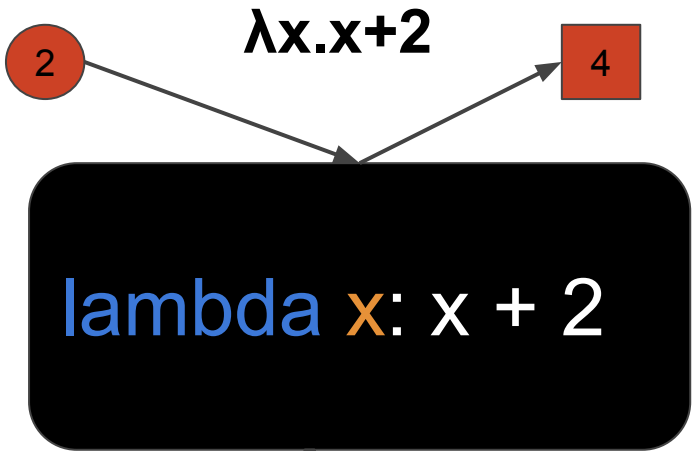

```
def mais_2(x):
    return x + 2
```

Aplicação de uma função simples



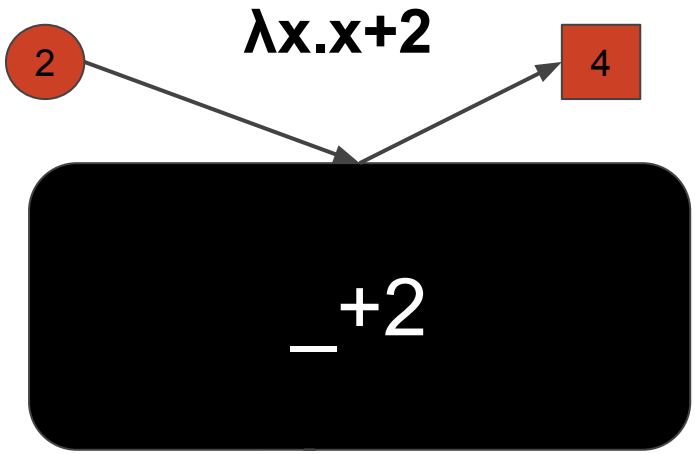

```
In [1]: def mais_2(x):
   ...:     return x + 2
   ...:

In [2]: mais_2(2)
Out[2]: 4
```

Aplicação de uma função anônima [0]



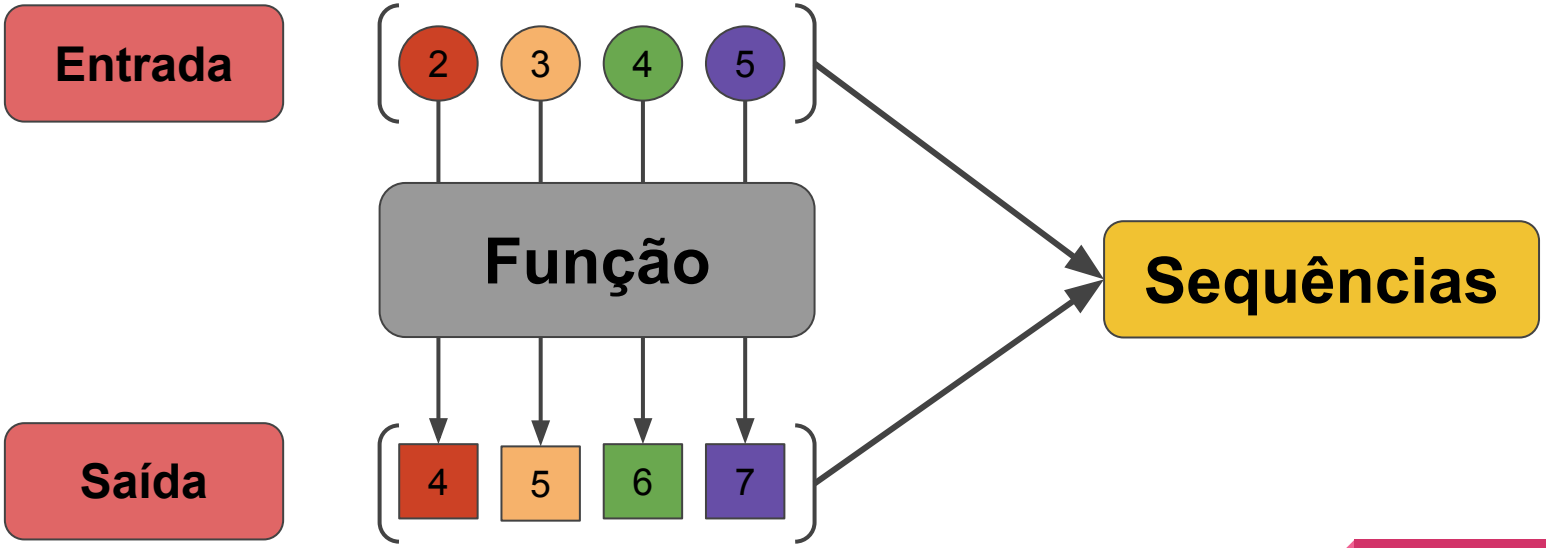

```
In [1]: (lambda x: x + 2)(2)
Out[1]: 4
```

Aplicação de uma função anônima [1]

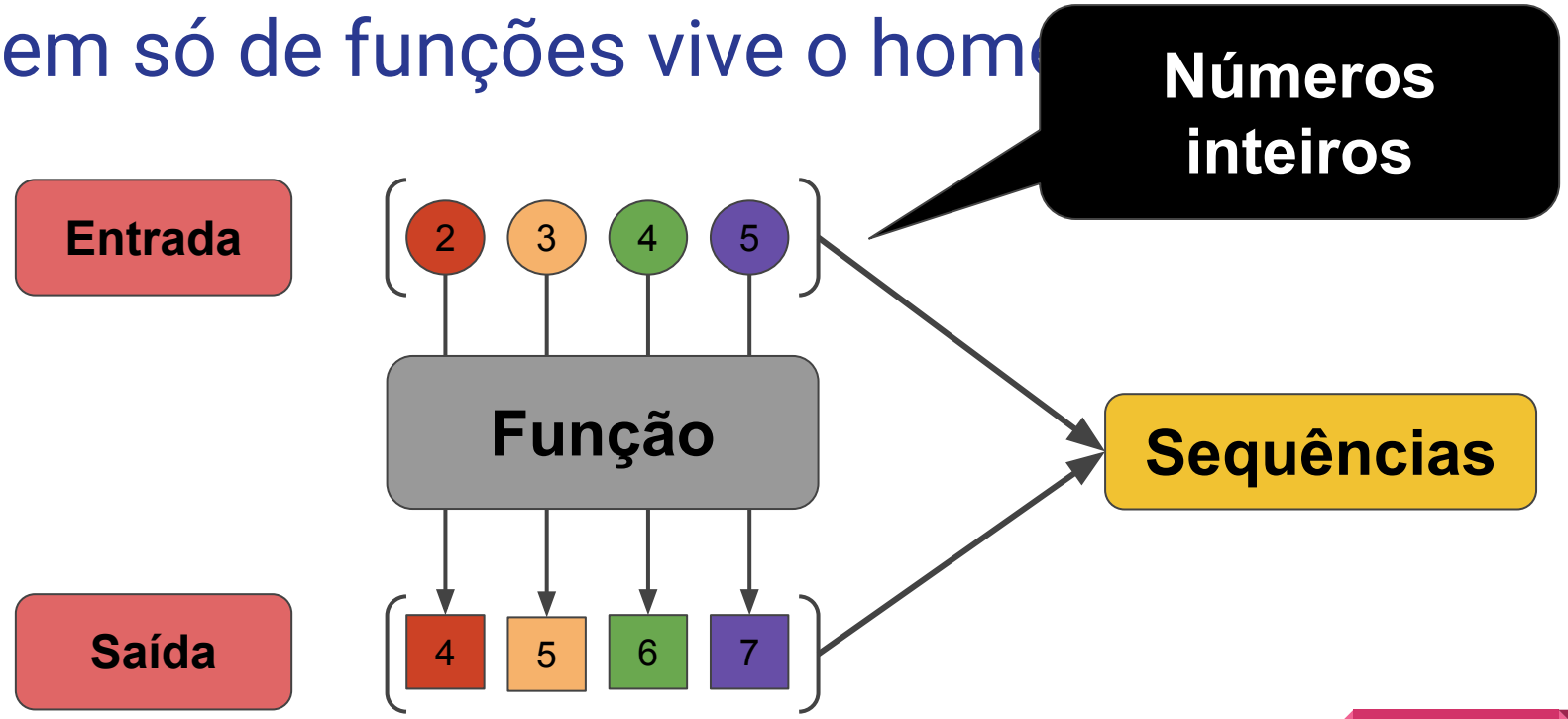



```
In [1]: from fn import _

In [2]: (_+2)(2)
Out[2]: 4
```

Nem só de funções vive o homem



Nem só de funções vive o hom



Tipos de sequência (as básicas)

- Strings
- Tuplas
- Listas
- Conjuntos
- Dicionários

Todas são iteráveis

Tipos de sequência (as básicas)

- Strings
- Tuplas
- Listas
- Conjuntos
- Dicionários

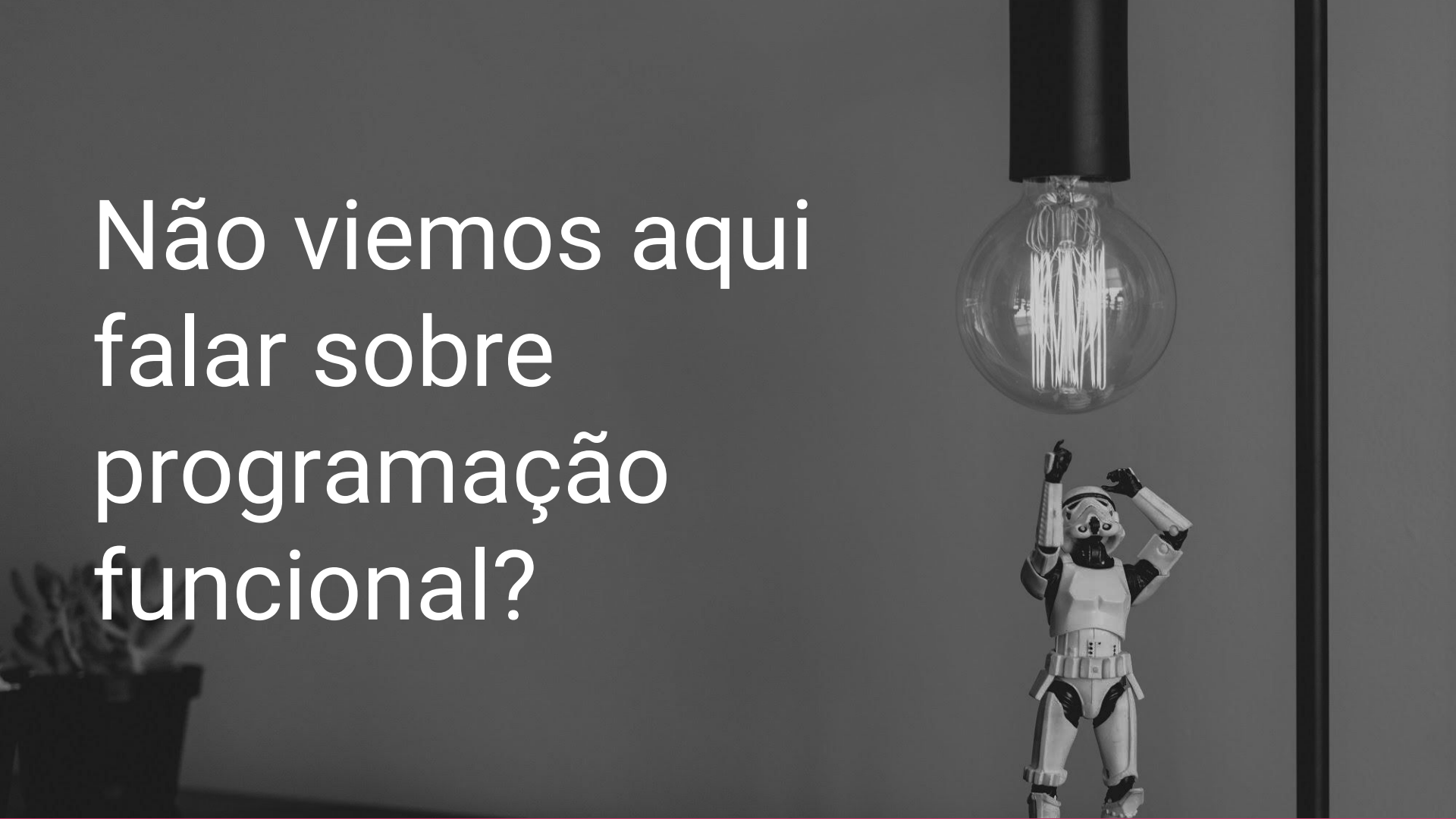
```
In [5]: '__iter__' in dir(tuple())
Out[5]: True

In [6]: '__iter__' in dir(list())
Out[6]: True

In [7]: '__iter__' in dir(dict())
Out[7]: True

In [8]: '__iter__' in dir(set())
Out[8]: True

In [9]: '__iter__' in dir(str())
Out[9]: True
```

Como iterar?

```
In [11]: for x in [1, 2, 3, 4, 5]:
    ...:     print(x)
    ...:
    ...:

1
2
3
4
5
```

Literalmente passear entre os elementos da lista

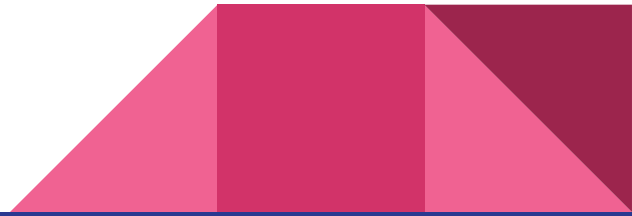
Não viemos aqui falar sobre programação funcional?



Funções de redução

Funções de redução são funções que recebem um iterável e retornam um único valor de resposta (nem sempre)

- any()
- all()
- len()
- sum()



Funções de mapeamento

Funções de redução são funções que recebem um iterável (nem sempre) e retornam outro iterável

- reversed()
- enumerate()
- zip()
- ~~map()~~

Lazy evaluation - Avaliação preguiçosa

São retornos (as vezes de funções) que só são computados quando chamados

Comumente são chamados pelo método __next__() pela função next()

Iterando com map

a função map recebe uma função e um iterável e retorna um iterável:

```
map(func, iter)
```

Funções de ordem superior

Funções de ordem superior são funções que recebem e/ou retornam funções:

- map()
- max()
- min()
- iter()
- sorted()
- filter()

Closures

Funções que podem ser ‘instanciadas’ e referenciar uma função interna:

```
In [1]: def externa(arg_1):
   ...:     def interna(arg_2):
   ...:         return arg_1 + arg_2
   ...:     return interna
   ...:

In [2]: closure = externa(2)

In [3]: closure(2)
Out[3]: 4

In [4]: closure(4)
Out[4]: 6
```

XOXO

Dúvidas?

github.com/z4r4tu5tr4/python-funcional
mendesxeduardo@gmail.com