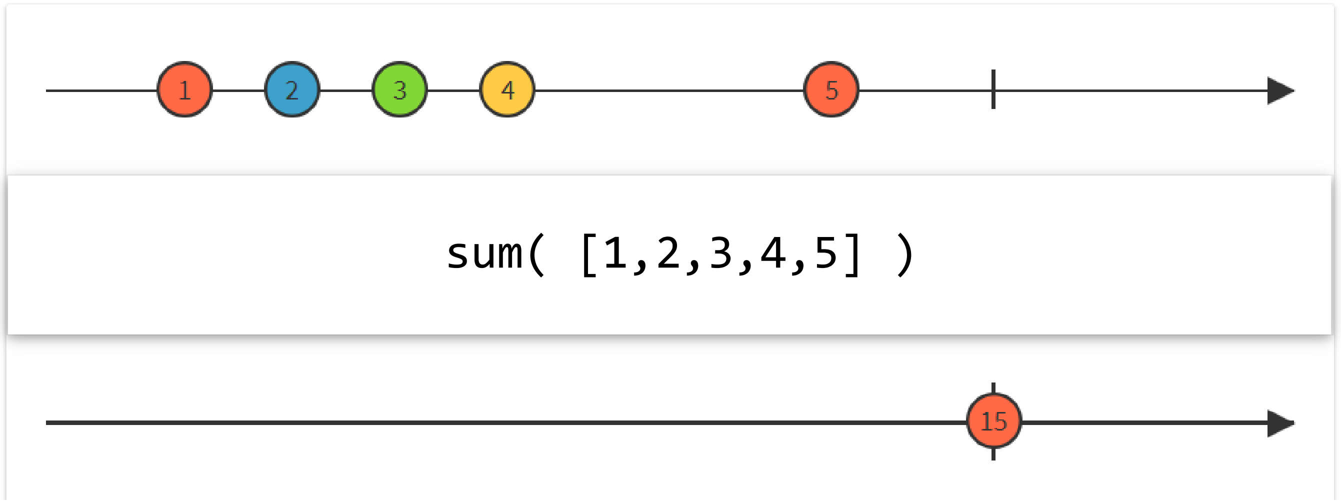
On lists and lambda

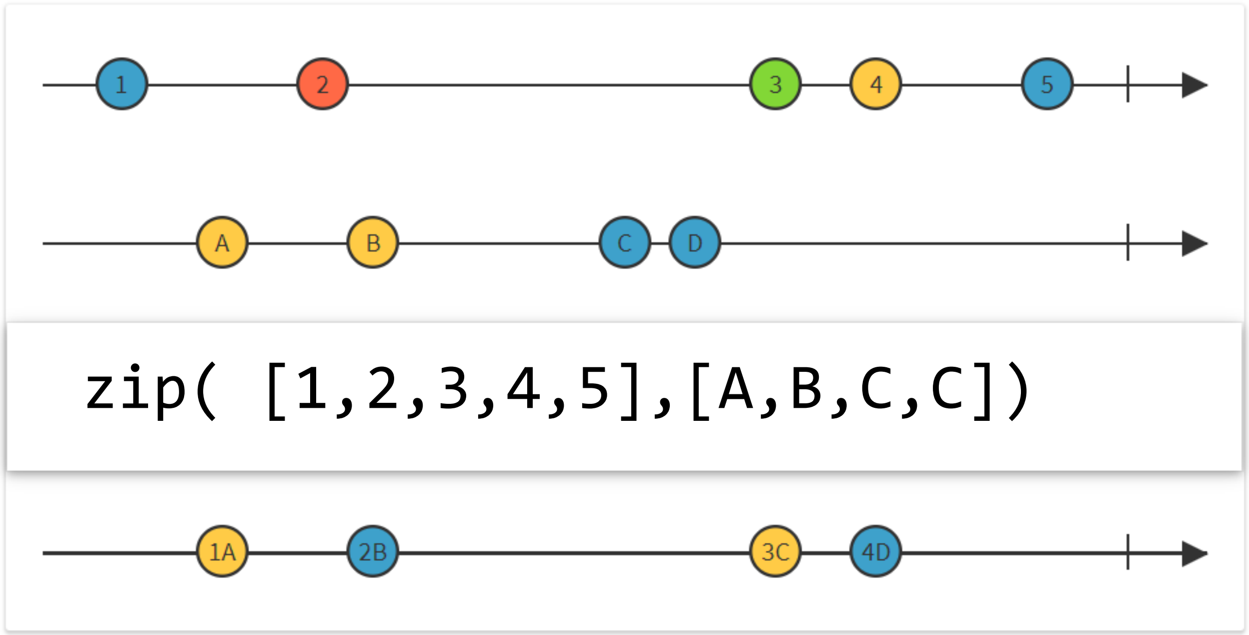
support

universidade de aveiro

Don’t reinvent the wheel

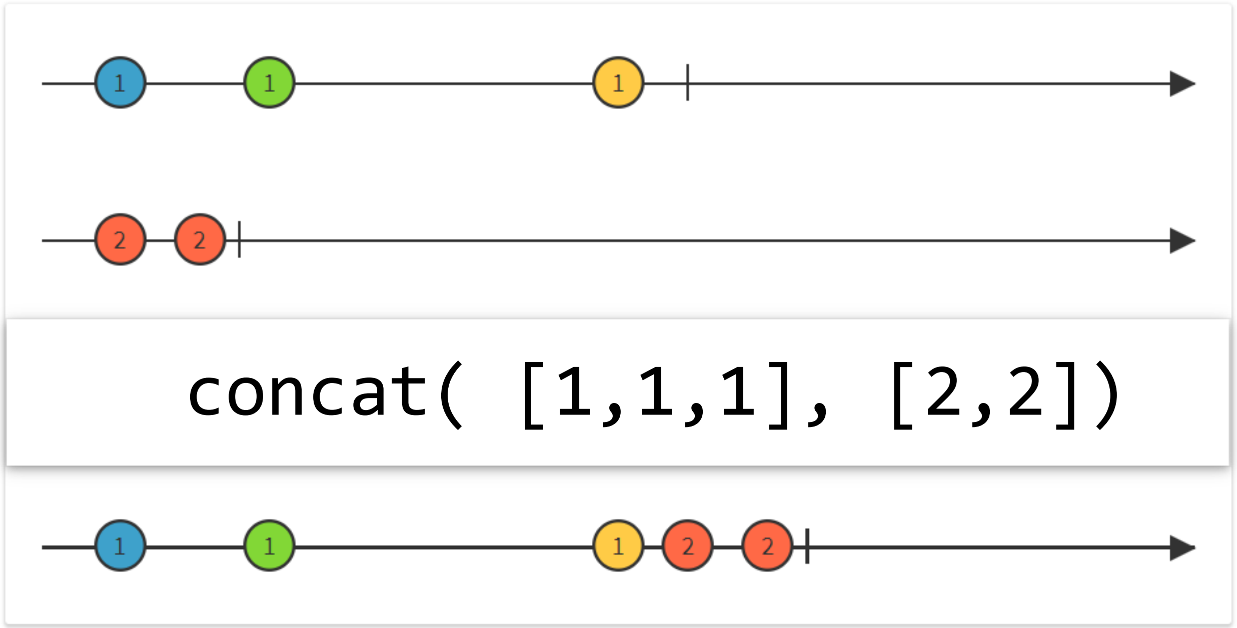


Don’t reinvent the wheel



```
zip( [1,2,3,4,5],[A,B,C,C])
```

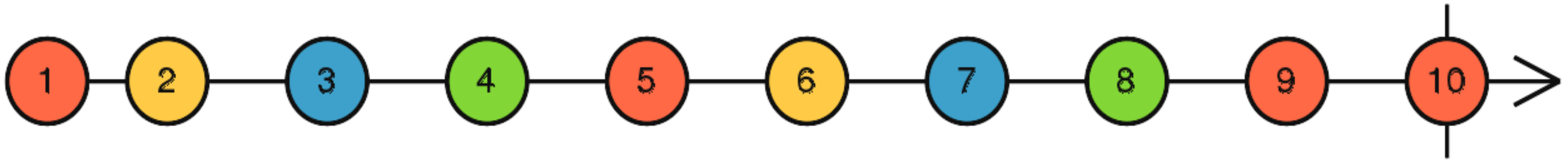
Don’t reinvent the wheel



universidade de aveiro

Don’t reinvent the wheel

```
range( 1 , 11 )
```



Lambda

These functions are called anonymous because they are not declared in the standard manner by using the *def* keyword. You can use the *lambda* keyword to create small anonymous functions.

- Lambda forms can take any number of arguments but return just one value in the form of an expression. They cannot contain commands or multiple expressions.
- An anonymous function cannot be a direct call to print because lambda requires an expression
- Lambda functions have their own local namespace and cannot access variables other than those in their parameter list and those in the global namespace.

https://www.tutorialspoint.com/python/python_functions.htm
https://realpython.com/python-lambda/

Lambda: ~ unnamed function

```
>>> sum = lambda x, y : x + y
>>> sum(3,4)
7
>>>
```

```
>>> def sum(x,y):
...     return x + y
...
>>> sum(3,4)
7
>>>
```

https://www.python-course.eu/python3_lambda.php

lambda

```
lambda arguments: expression
```

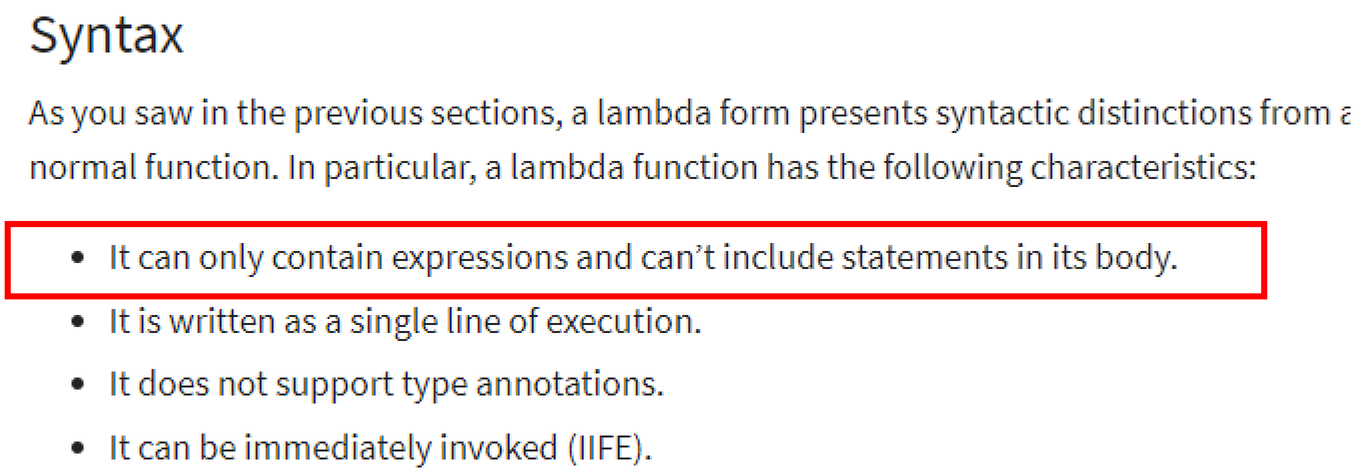
- This function can have any number of arguments but only one expression, which is evaluated and returned.
- One is free to use lambda functions wherever function objects are required.
- You need to keep in your knowledge that lambda functions are syntactically restricted to a single expression.
- It has various uses in particular fields of programming besides other types of expressions in functions.

```
# Python code to illustrate cube of a number
# showing difference between def() and lambda().
def cube(y):
    return y*y*y;

g = lambda x: x*x*x
print(g(7))

print(cube(5))
```

```
(lambda x: x * x)(3)
```

https://www.w3schools.com/python/python_lambda.asp
https://realpython.com/python-lambda/
https://www.geeksforgeeks.org/python-lambda-anonymous-functions-filter-map-reduce/

lambda



Syntax

As you saw in the previous sections, a lambda form presents syntactic distinctions from a normal function. In particular, a lambda function has the following characteristics:

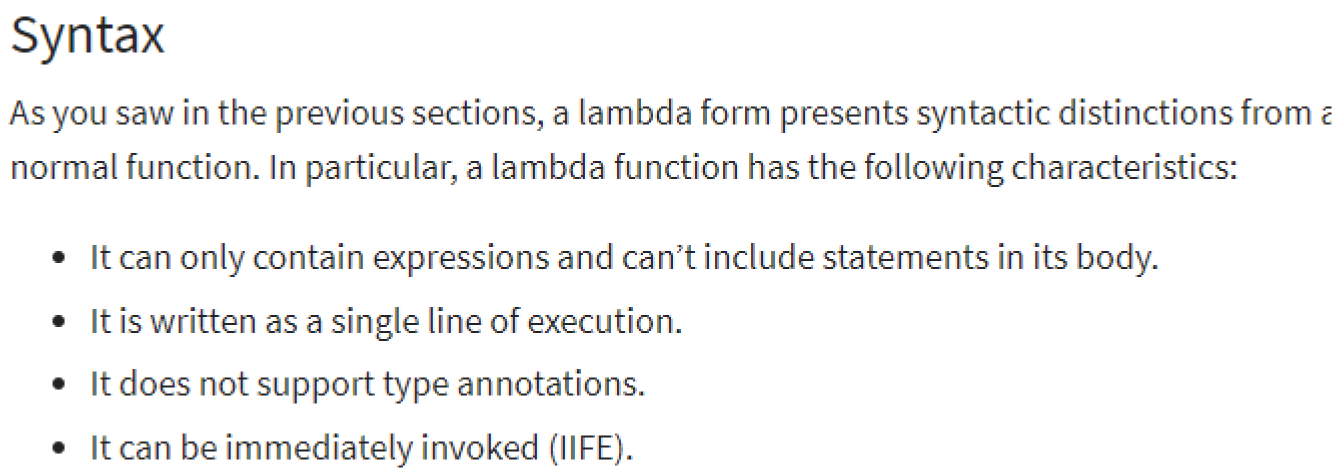
- It can only contain expressions and can't include statements in its body.
- It is written as a single line of execution.
- It does not support type annotations.
- It can be immediately invoked (IIFE).

```
intervalo1 = lambda a, b : (b,a) if a>b else  (a,b)
```

```
intervalo1 = lambda a, b : if a>b : (b,a) else: (a,b)
```
❌

https://realpython.com/python-lambda/

lambda



Syntax

As you saw in the previous sections, a lambda form presents syntactic distinctions from
normal function. In particular, a lambda function has the following characteristics:

- It can only contain expressions and can't include statements in its body.
- It is written as a single line of execution.
- It does not support type annotations.
- It can be immediately invoked (IIFE).

```
intervalo1 = lambda a, b : (b,a) if a>b else  (a,b)
```

```
intervalo1 = lambda a, b : if a>b : (b,a) else: (a,b)
```

https://realpython.com/python-lambda/

universidade de aveiro

Just test

```
my_list = [1, 5, 4, 6, 8, 11, 3, 12]

new_list = list(filter(lambda x: (x%2 == 0) , my_list))

# Output: [4, 6, 8, 12]
print(new_list)
```

```
def myfunc(n):
  return lambda a : a * n

mydoubler = myfunc(2)
mytripler = myfunc(3)

print(mydoubler(11))
print(mytripler(11))
```

```
my_list = [1, 5, 4, 6, 8, 11, 3, 12]

new_list = list(map(lambda x: x * 2 , my_list))

# Output: [2, 10, 8, 12, 16, 22, 6, 24]
print(new_list)
```

https://www.programiz.com/python-programming/anonymous-function
https://www.w3schools.com/python/python_lambda.asp

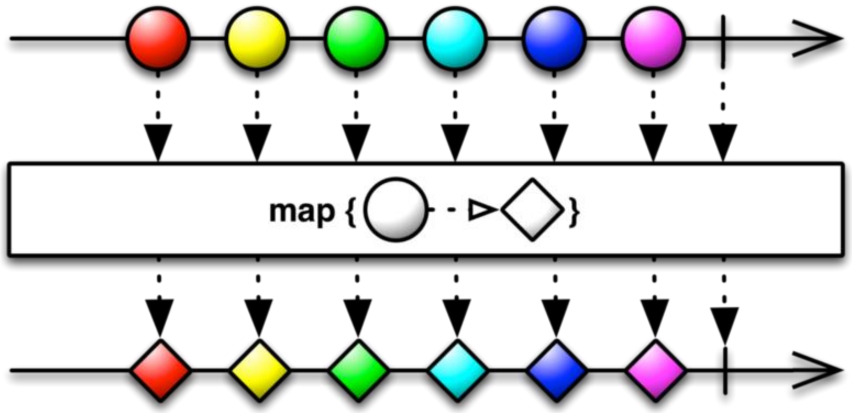
Lambda and parameters

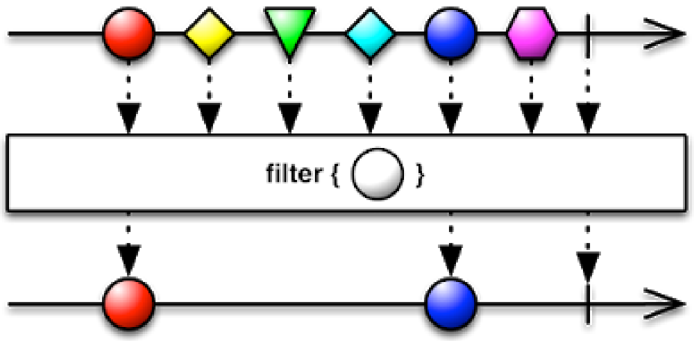
```
>>> (lambda x, y, z: x + y + z)(1, 2, 3)
6
>>> (lambda x, y, z=3: x + y + z)(1, 2)
6
>>> (lambda x, y, z=3: x + y + z)(1, y=2)
6
>>> (lambda *args: sum(args))(1,2,3)
6
>>> (lambda **kwargs: sum(kwargs.values()))(one=1, two=2,
three=3)
6
>>> (lambda x, *, y=0, z=0: x + y + z)(1, y=2, z=3)
6
```

Key Functions

- higher-order functions
  - receives a function that can be a lambda.
    - directly influences the algorithm driven
  - that take a parameter key as a named argument.
- Here are some key functions:
  - Map, filter, reduce
  - sort(): list method
  - sorted(), min(), max(): built-in functions
  - all()

Map, Filter, Reduce

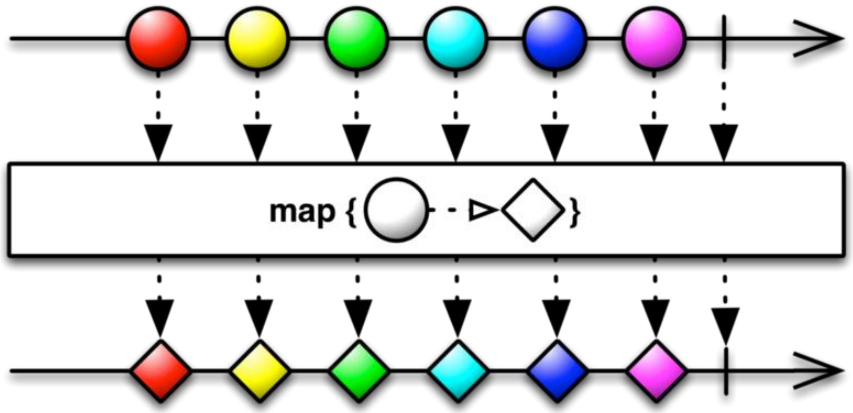


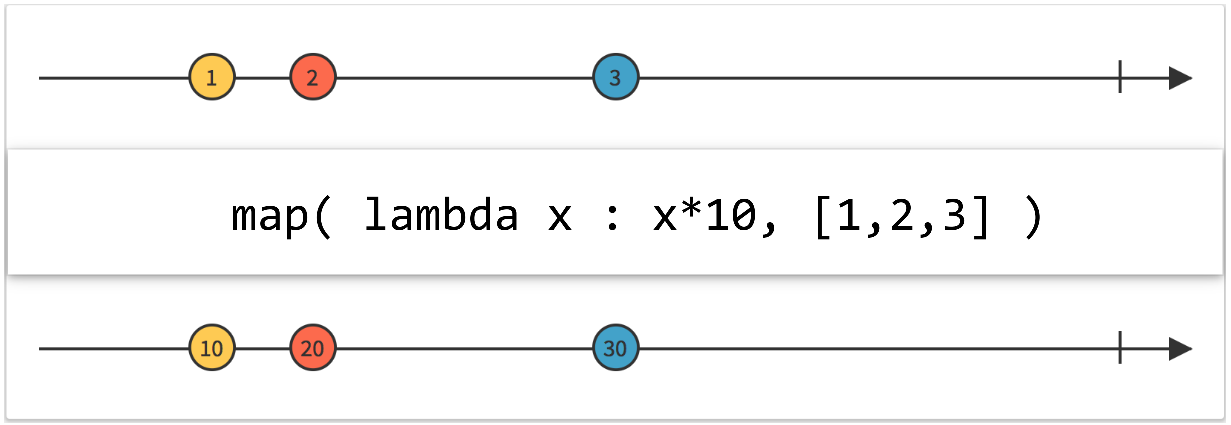


https://www.learnpython.org/en/Map,_Filter,_Reduce
https://www.python-course.eu/python3_lambda.php

map

map(function_object, iterable1, iterable2,...)





```
map( lambda x : x*10, [1,2,3] )
```

map

```
def multiply2(x):
  return x * 2

map(multiply2, [1, 2, 3, 4])
# Output [2, 4, 6, 8]

list_a = [1, 2, 3]
list_b = [10, 20, 30]

map(lambda x, y: x + y, list_a, list_b)
# Output: [11, 22, 33]
```

https://medium.com/better-programming/lambda-map-and-filter-in-python-4935f248593

map

```
def mymap( f , l ):
  res=[]
  for x in l :
    res.append( f(x) )
  return res
(NOTE: not exactly the same but outputs the same results)

l=[1, 2, 3, 4]

print( list( map( lambda x:x**2 , l ) ) )
print( list( mymap( lambda x:x**2 , l ) ) )
```

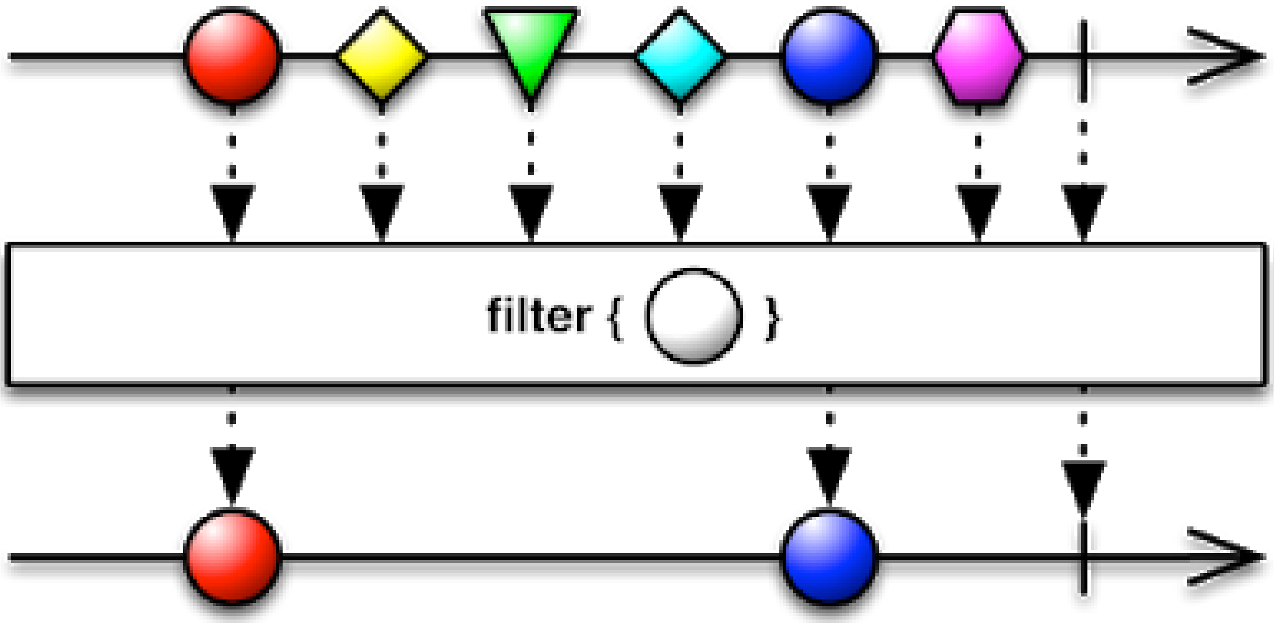
Map: Multiple lists

```
list_a = [1, 2, 3]
list_b = [10, 20, 30]
map(lambda x, y: x + y, list_a, list_b)
 # Output: [11, 22, 33]
```

https://medium.com/@happymishra66/lambda-map-and-filter-in-python-4935f248593

filter

filter(function_object, iterable)



filter

```
a = [1, 2, 3, 4, 5, 6]
filter(lambda x : x % 2 == 0, a) # Output: [2, 4, 6]

dict_a = [{'name': 'python', 'points': 10}, {'name': 'java',
'points': 8}]
filter(lambda x : x['name'] == 'python', dict_a) # Output:
[{'name': 'python', 'points': 10}]


list_a = [1, 2, 3, 4, 5]

filter_obj = filter(lambda x: x % 2 == 0, list_a) # filter
object <filter at 0x4e45890>
even_num = list(filter_obj) # Converts the filer obj to a
list
print(even_num) # Output: [2, 4]
```

https://medium.com/better-programming/lambda-map-and-filter-in-python-4935f248593

filter

```
def myfilter( f , l ):
  res=[]
  for x in l :
    if f(x) :
      res.append( x )
  return res

l=[1, 2, 3, 4]

print( list( filter( lambda x:x%2==0 , l )) )
print( list( myfilter( lambda x:x%2==0 , l)) )
```

Printing map & filters results…

```
map_output = map(lambda x: x*2, [1, 2, 3, 4])
print(map_output) # Output: map object: <map object at
0x04D6BAB0>
list_map_output = list(map_output)

print(list_map_output)
# Output: [2, 4, 6, 8]



list_a = [1, 2, 3, 4, 5]
filter_obj = filter(lambda x: x % 2 == 0, list_a) # filter
object <filter at 0x4e45890>
even_num = list(filter_obj) # Converts the filter obj to a list
print(even_num)
# Output: [2, 4]
```

https://medium.com/@happymishra66/lambda-map-and-filter-in-python-4935f248593

Min, max

```
# Python code explaning min() and max()
l = ["ab", "abc", "bc", "c"]

print(max(l, key = len))
print(min(l, key = len))
# you can also write in this form
print(max(l, key = lambda element:len(element)))
#ouput
abc
c
abc
```

https://www.geeksforgeeks.org/use-of-min-and-max-in-python/

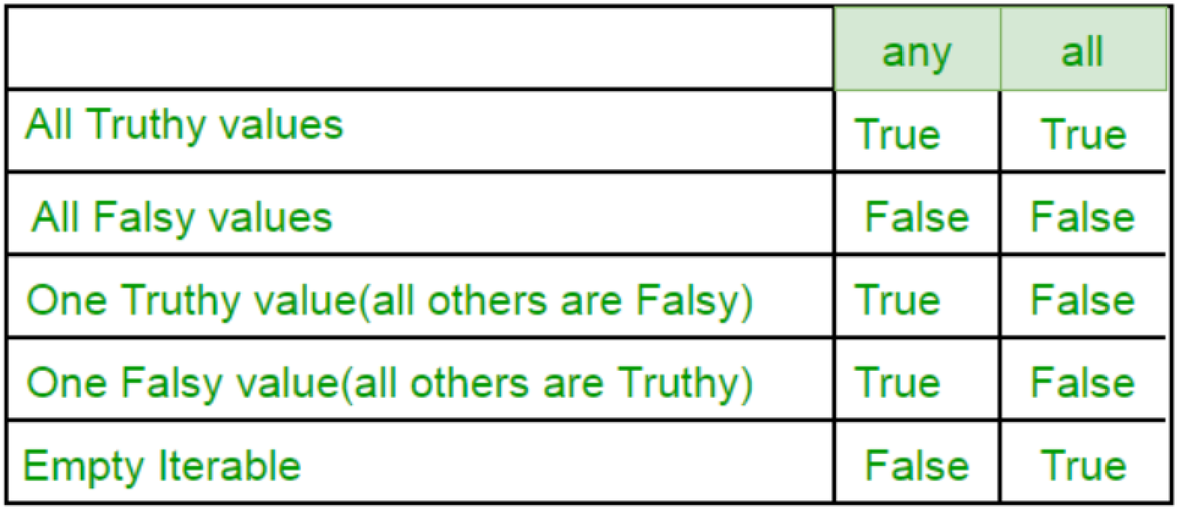
Any, all

```
# Here all the iterables are True so all
# will return True and the same will be printed
print (all([True, True, True, True])) # → True
# Here the method will short-circuit at the
# first item (False) and will return False.
print (all([False, True, True, False])) # →False

# This statement will return False, as no
# True is found in the iterables
print (all([False, False, False]))  # →False
```

https://www.geeksforgeeks.org/any-all-in-python/

Any, all

```
# Here all the iterables are True so all
# will return True and the same will be printed
print (all([True, True, True, True])) # → True
# Here th
# first i
print (al


# This st
# True is
print (all([False, False, False]))  # →False
```

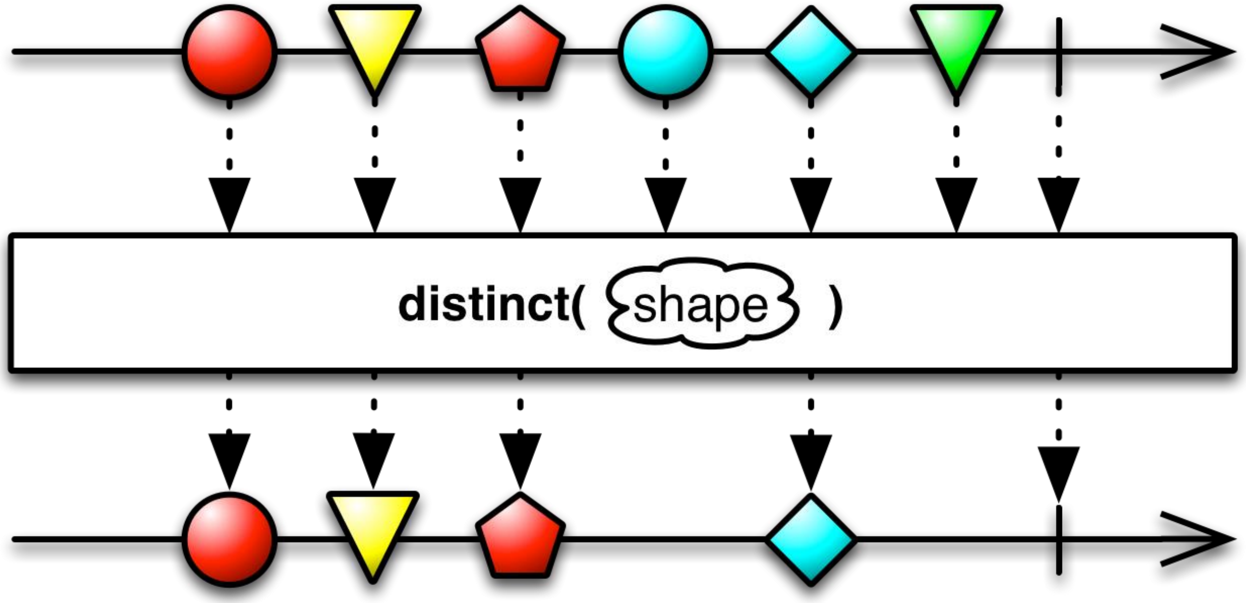


	any	all
All Truthy values	True	True
All Falsy values	False	False
One Truthy value(all others are Falsy)	True	False
One Falsy value(all others are Truthy)	True	False
Empty Iterable	False	True

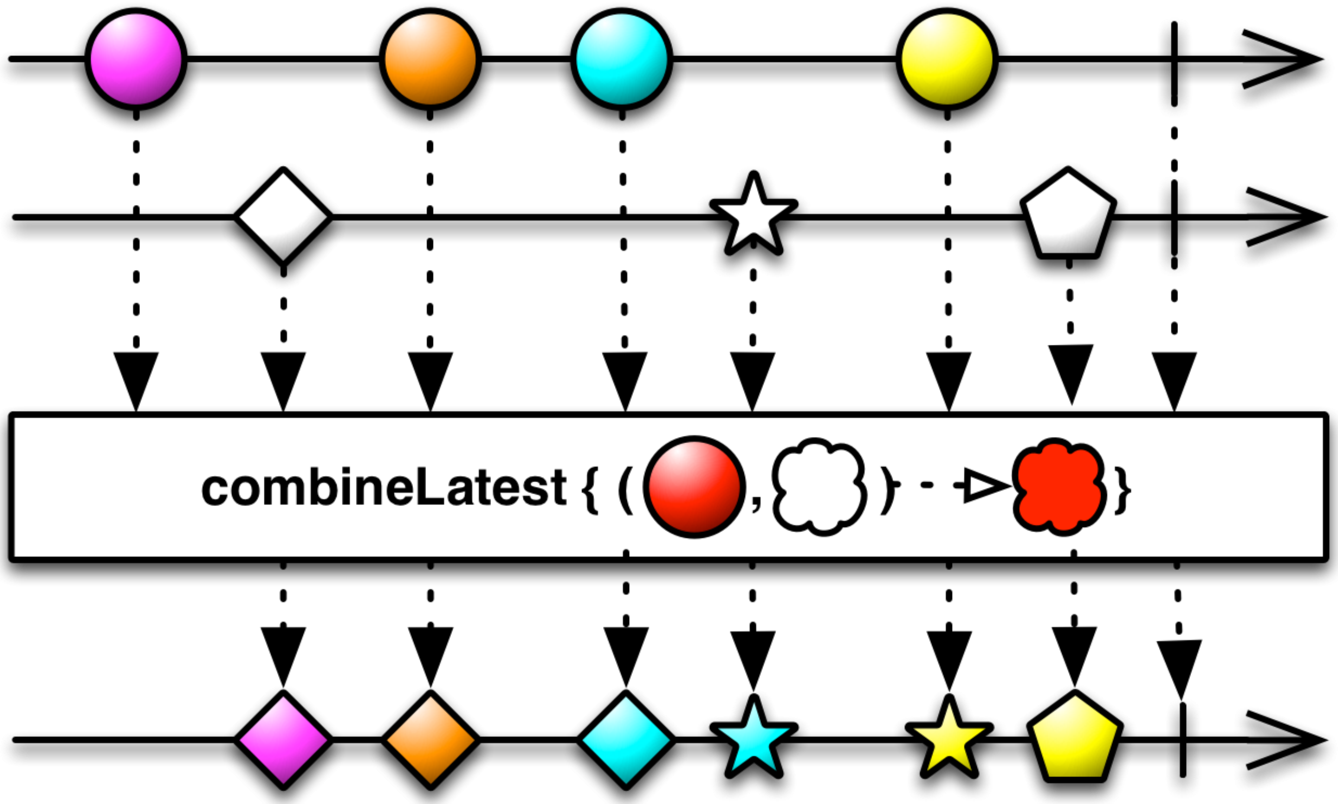
https://www.geeksforgeeks.org/any-all-in-python/

The END

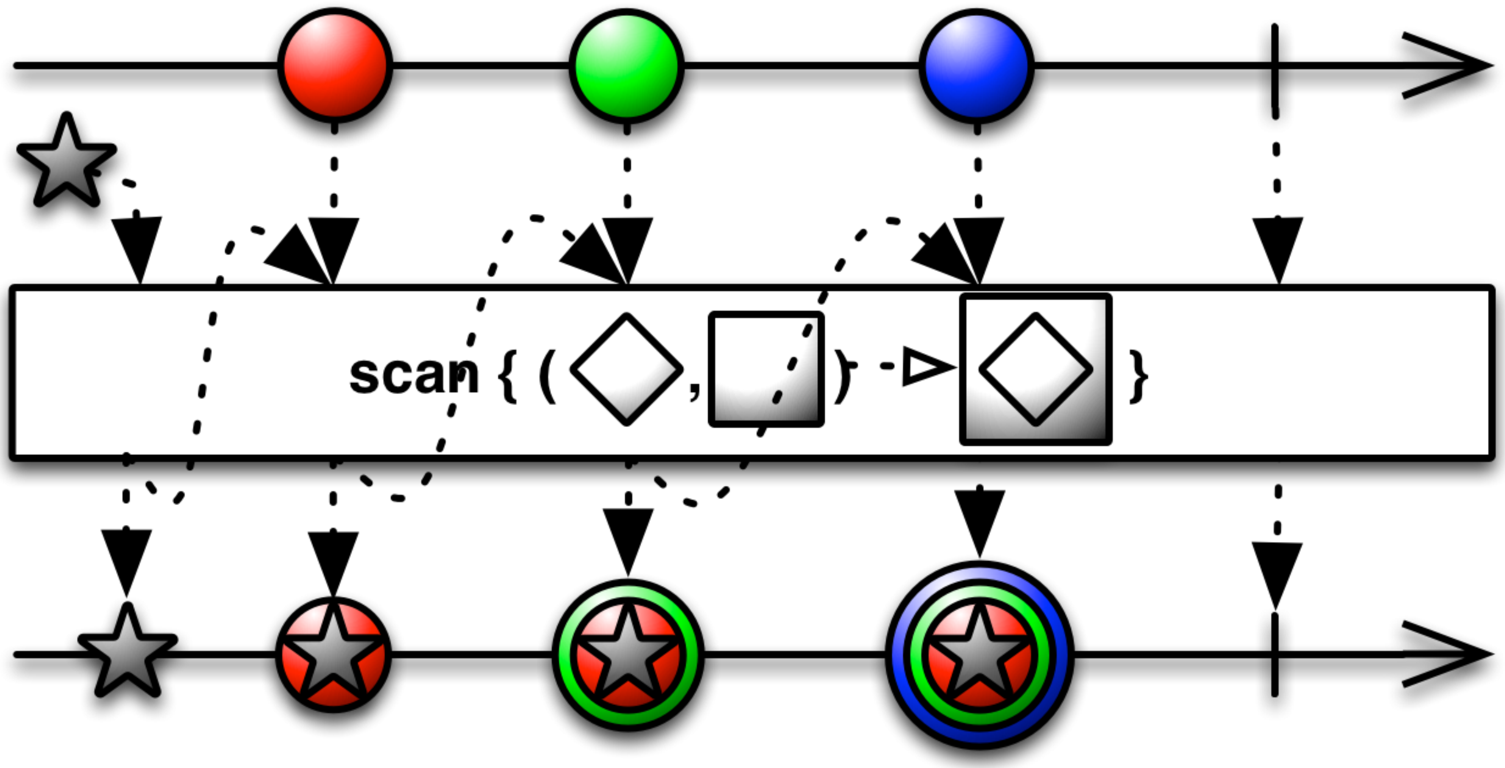
Challenges: try to implement…



https://github.com/Froussios/Intro-To-RxJava/blob/master/Part%202%20-%20Sequence%20Basics/2.%20Reducing%20a%20sequence.md



https://pursuit.purescript.org/packages/purescript-rx-observable/1.1.3/docs/RxJS.Observable


scan { (,) }

Support on Lists and Lambda