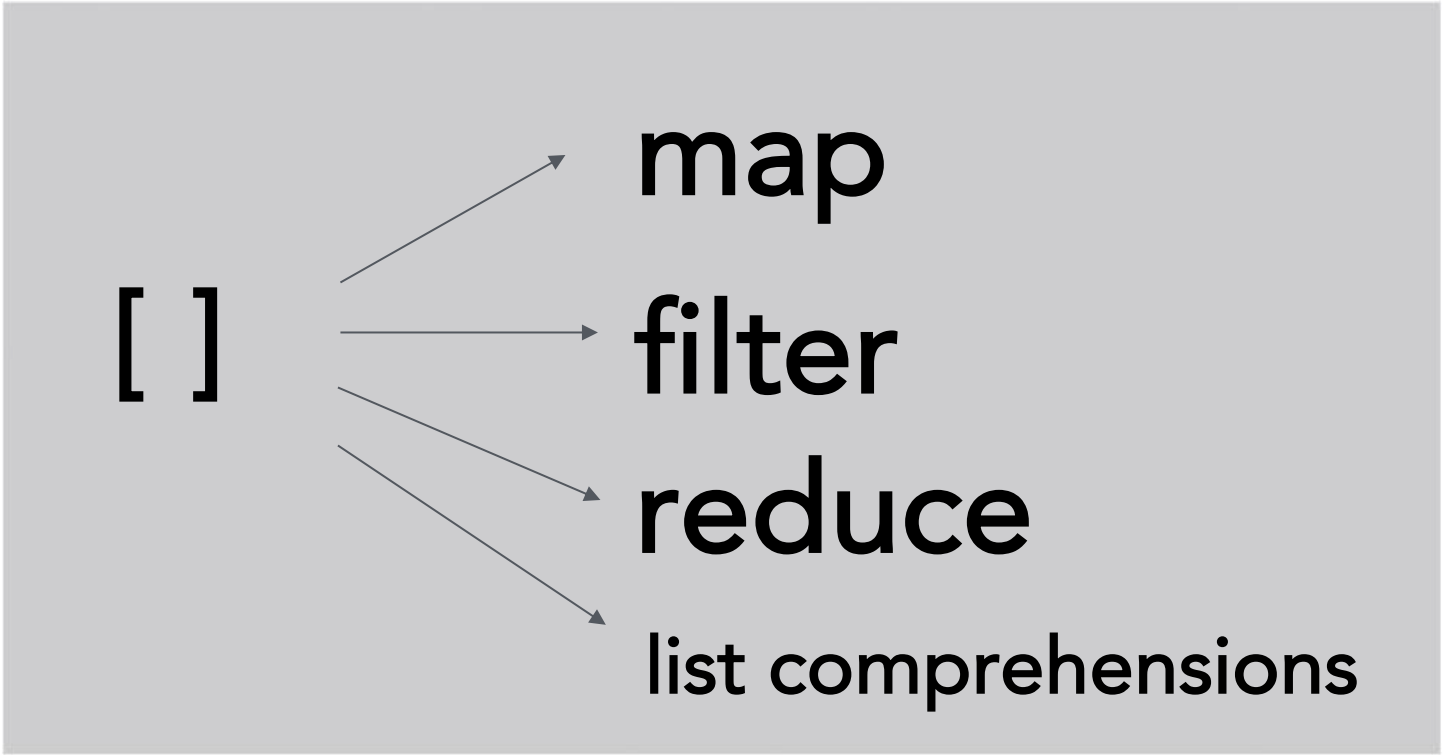
Working with Lists “Functionally”



Python’s “map” Function

“map” is used to convert each element in an array to some other value

Python's "map" Function

```
doubled_numbers = []

for number in numbers:
  doubled = number * 2
  doubled_numbers.append(doubled)
```

Python's "map" Function

```
numbers = [1, 2, 3, 4, 5]

doubled_numbers = map(double, numbers)
```

Python’s “filter” Function

“filter” is used to get all the elements in an array that fit certain criteria

Python’s “filter” Function

```
filter(someFunc, list)
```

Lambdas in Python

```
def someFunc(x):
   …

filter(someFunc, list)
```

Lambdas in Python

Lambdas are simply one-line functions that can be defined inside other expressions

Lambdas in Python

```
lambda x, y: x + y
```

Python’s “reduce” Function

“reduce” takes all the elements in an array and combines them into a single value

Python's "reduce" Function

```
numbers = [ 1, 2, 3, 4, 5, 6, 7 ]

0 <- this is the starting value
0 + 1 <- add the first number in the array
1 + 2 <- add the second number in the array
3 + 3 <- add the third number in the array
...
```

Python's "reduce" Function

```
numbers = [ 1, 2, 3, 4, 5, 6, 7 ]

1 <- this is the starting value
1 * 1 <- multiply by the first number
1 * 2 <- multiply by the second number
2 * 3 <- multiply by the third number
6 * 4 <- multiply by the fourth number
…
```

Python's "reduce" Function

```
reduce(some_function, my_list)
```

```
def some_function(acc, element):
    ...
```

Python's "reduce" Function

```
def get_sum(acc, element):
  return acc + element
```

Python's "reduce" Function

```
reduce(get_sum, my_list, 0)
```

Python's "reduce" Function

The "accumulator" function:

```
myArray.reduce(
   (acc, x) => acc + x
);
```

Python's "reduce" Function

```
myArray.reduce(
   (acc, x) => acc + x,
   0 // the starting value
);
```