Higher order functions in Python : sorted, map, reduce, filter

[Python](https://iamit.in/blog/Higher-order-functions-in-Python-sorted-map-reduce-filter/tags/#Python)

Python has a pretty good support for Higher order functions, we are going to learn their implementation here :bulb: .

What are higher order functions?

***“Functions that take a function as an argument or return a function.”***

Example:

def foo(f):

f() # calling the function that is passed

def bar():

print "Hello bar"

foo(bar) # function bar is passed to function foo.

Hello bar

Functions and methods are [first-class citizen](http://en.wikipedia.org/wiki/First-class_citizen) (objects) in Python, so if we want to pass a function as an argument to another function, we can just treat it as any other object.

It’s worth noting that in some languages functions and objects are distinct, but in python functions are implemented as objects (That also means functions have attributes).

sorted

[sorted](https://wiki.python.org/moin/HowTo/Sorting) does pretty much the same thing what you expected.

# sorting numbers

In [0]: sorted([9, 5, 7])

Out[0]: [5, 7, 9]

# sorting alphabetically

In [0]: sorted(['foo', 'bar'])

Out[0]: ['bar', 'foo']

Using Key :key:

Well you can also ***customize*** the sorting also, the parameter key. The key needs to be a simple function the takes a single value that tells python the value to use in sorting it.

In [0]: sorted([1, -3, 2]) # sorting normally

Out[0]: [-3, 1, 2]

In [0]: sorted([1, -2, 0], key=abs) # sorting by absolute values

Out[0]: [0, 1, -2]

Note: abs is the absolute value function. i.e. abs(-3) = 3

map

*Takes an input iterable of values and return a list with different values. Same order, same length, but mapped via a function.*

map(function, iterable) map takes a function as the first parameter and then an iterable containing data.

In [0]: def twice(x):

return 2\*x

In [1]: map(twice, [1, 2, 3, 4, 5])

Out[1]: [2, 4, 6, 8, 10]

filter

*filter takes a function and an iterable and produces an output list of every item on the input list that passes a test*.

In [0]: def greater\_than\_5(x):

return x > 5

In [1]: filter(greater\_than\_5, range[10])

Out[1]: [6, 7, 8, 9]

***Special case***: When None is passed instead of function, it will filter by Truthiness.

filter(None, [0, 2, 1, 3])

[2, 1, 3]

filter(None, ['a','', 'b' ])

['a', 'b']

reduce

reduce takes an iterable of input data and consumes it to come up with a single value.

*The function we have to write is different too - the function will take two values and typically reduce will consume the first two values from the iterable then consume one value at a time from the list using the return value from the previous call.*

In [3]: def add(x,y):

...: return x + y

...:

In [4]: reduce(add, [1,2,3,4])

Out[4]: 10