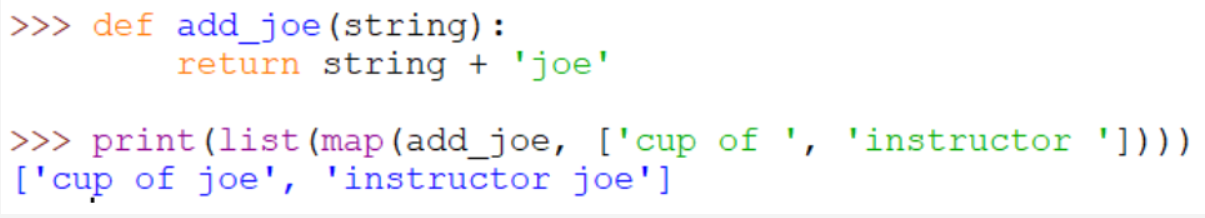
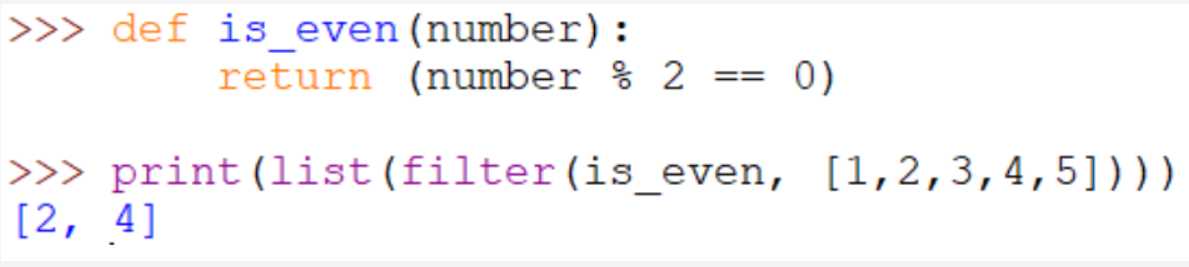
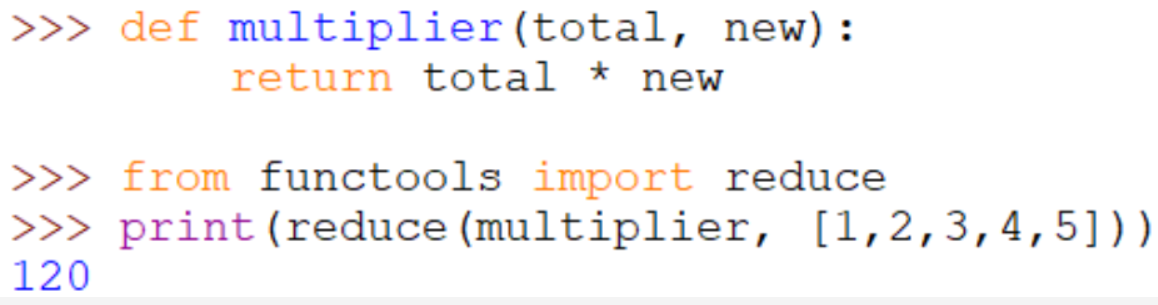
**Advanced Python Pointer: Map/Filter/Reduce**

Now that you’ve learned about lambda functions, let’s take a look at another functional programming concept: map/reduce (https://en.wikipedia.org/wiki/MapReduce). An early mechanism for performing Big Data calculations, map/reduce programming concepts have started to fall out of favor, though there are still modern applications of the logic used in Big Data scenarios today (such as Apache Spark). Furthermore, map/reduce techniques can be used to simplify code and to express certain operations in much cleaner fashions. The 3 main pillars of map/reduce in python are the builtin functions map(), filter(), and reduce().

map(function, list) : takes 2 arguments, a function and a list. It then proceeds to perform the function across all the elements of the list. The return value of the function represents a transformed version of the original element. map() returns a map object, that can easily be turned back into another list with the list() function. In a nutshell, map() is a way of performing some kind of transform on all the items in a list. The total number of elements in the new list should be equal to the total number of elements in the original list.  


filter(function, list) : like map, takes 2 arguments, a function and list. However, while map is designed to transform all the elements in a list, filter() is designed to winnow down the existing elements in the list. Rather than returning a transformed copy of an element, the filter() function should return a boolean, determining whether the element should be included in the new list. So the total number of elements in the new list after a filter may be less than the total number of elements in the original list. Similar to map, you will need to use the list() function to transform your filter() object.

reduce(function, list, initial=None) : takes 2 required arguments and 1 optional argument. reduce() works differently than map() and filter() in that the end result for a reduce() function is not a list. Reduce()’s job is to convert a list into a singleton. In both map() and filter(), the inner function that was being invoked only had 1 argument, the element that was being processed. Reduce()’s inner function, however, requires 2 arguments, the running total value and the new element in the list. The return value at the end of reduce is the final outcome. Optionally, you can add in initial value for the first iteration as well. One more thing to note about reduce, is that it is no longer a builtin keyword in python. Instead, you must import it from the functools module (don’t worry about the details on this – we'll cover them after the midterm – for now, just know the “from … import ...” line is required).



For more information and examples on map, filter and reduce, check out the writeup here: <https://stackabuse.com/map-filter-and-reduce-in-python-with-examples/>