**Underscore**

Objectives:

* Understand callbacks
* Practice anonymous functions
* Create a custom Python module

Your own custom Python Module!

Did you know that you can actually create your own custom python module similar to the [Underscore](http://underscorejs.org/) library in JavaScript? That may be hard to believe, but in truth, you know how to create significant Python modules of your own. To create a custom Python module, you will simply define several functions, or methods, into a single class! We'll talk about classes soon--for now you can use the code below and follow the pattern for calling the methods as shown below.

You will create the following methods from the underscore library as **methods** of the "\_" object. **Pay attention to what you have to change, in terms of parameters for each method as well as implementation.**

In each of the following methods, the first parameter, **self**, is implicitly passed (again, more on this in the next chapter). The only parameters you need to worry about for now are **iterable** and **callback**. **Iterable** will be the list being passed in, and **callback** will be the lambda function.

*class Underscore:*

*def map(self, iterable, callback):*

*# your code here*

*def find(self, iterable, callback):*

*# your code here*

*def filter(self, iterable, callback):*

*# your code*

*def reject(self, iterable, callback):*

*# your code*

*# you just created a library with 4 methods!*

*# let's create an instance of our class*

*\_ = Underscore() # yes we are setting our instance to a variable that is an underscore*

*evens = \_.filter([1, 2, 3, 4, 5, 6], lambda x: x % 2 == 0)*

*# should return [2, 4, 6] after you finish implementing the code above*

In the code above, you just created your own custom Python module/library that others can use! How can others use the methods in your library? By calling the properties stored in the class you defined (e.g. \_.map(), \_.find(), etc).

Your assignment is to implement the 4 methods above using callbacks. You will have to modify the 4 methods to take in a list and a callback. A callback is simply a function that is passed as an argument, to be executed by the function to which it is being passed. Just as we are able to pass numbers, lists, strings, etc. when making a function call, we can also pass functions! That means we do not invoke the function right away, but rather pass the function by using the name only (i.e. not including the ()). In the following examples, we are specifically passing lambda functions:

*\_.map([1,2,3], lambda x: x\*2) # should return [2,4,6]*

*\_.find([1,2,3,4,5,6], lambda x: x>4) # should return the first value that is greater than 4*

*\_.filter([1,2,3,4,5,6], lambda x: x%2==0) # should return [2,4,6]*

*\_.reject([1,2,3,4,5,6], lambda x: x%2==0) # should return [1,3,5]*

One important concept that we want you to learn through this assignment is **how to pass data to and from callbacks.** You pass data **to a callback with a *parameter*** and you pass data **from the callback back to the parent function with a *return*.** While you are going through this assignment pay close attention to this relationship.

To understand what each method does, please refer to the [underscore library](http://underscorejs.org/). **Note that your method does not have to be as robust; you just need to get the base functionality working. For most methods, you will only have the list and a lambda as parameters, and for the lambda you will pass in each element and potentially a "memo" also known as a "collector".**

* Top of Form
* Complete the map method
* Complete the find method
* Complete the filter method
* Complete the reject method

Bottom of Form