University of Toronto
CSC148, Winter 2017

Recursion Exercises

```
def sum_list(L):
    """
    Return the sum of all ints in L.

    @param int|list[int|list[...]] L: possibly-nested list of ints, finite depth

    >>> sum_list([1, [2, 3], [4, 5, [6, 7], 8]])
    36
    >>> sum_list([])
    0
    """

    if isinstance(L, list):
        return sum([sum_list(x) for x in L])
    else:
        return L
```

(although sum([]) is 0 too)

```
new_list = []
for x in L:
    L.append(sum_list(x))
return sum(new_list)
```

1. What helper ~~methods~~ functions does this function call?

   sum → this already sums ints in a list
   isinstance

2. So far, we haven't confirmed that the function works in any cases. Trace this call: `sum_list(27)`

   isinstance is False, go straight to else block
   + return 27

3. Complete the following trace of this call: `sum_list([4, 1, 8])`

```
sum_list([4, 1, 8]) --> sum( [ sum_list(4), sum_list(1), sum_list(8) ] )
how many? 4         --> sum( [  4 , 1 , 8 ] )
total.              --> 13
```

4. Trace this call: `sum_list([4])`

   → sum([sum_list(4)])
   → sum([4])
   → 4

   replace each function call with return value

5. Trace this call: `sum_list([])`

   Q: how many calls to sum_list are made in total for this initial call? ⇒ 1 (just the initial one)
   → sum([ ])
   → 0

6. Trace this call: `sum_list([4, [1, 2, 3], 8])`

7. Trace this call: `sum_list([[1, 2, 3], [4, 5], 8])`

8. Trace this call: `sum_list([1, [2, 2], [2, [3, 3, 3], 2]])`

9. Trace this call: `sum_list([1, [2, 2], [2, [3, [4, 4], 3, 3], 2]])`

10. Are you a believer yet?