University of Toronto
CSC148, Winter 2015

# Recursion Exercises

```
def sum_list(L):
    '''
    (arbitrarily nested list of int or int) -> int

    Return the sum of all ints in L.
    >>> sum_list([1, [2, 3], [4, 5, [6, 7], 8]])
    36
    '''

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

or L itself if L is an int

I have confirmed that sum_list works for:

- int (depth 0)
- list of int, even empty or singleton ones (depth 1)
- list of depth 2
- lists of depth 3
- lists of depth 4
- lists of depth n if it works for depths 0 – n-1

1 min

1. What helper methods does this function call?

sum, isinstance, sum_list

1 min

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

sum_list(27) → 27

2 min

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)  ] )
                    --> sum( [   4,  1,  8])
                    -->   13
```

2 min

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

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

5. Trace this call: sum_list([])

sum_list([]) → sum([sum_list(x) for x in []])
→ sum([])
→ 0

4 min

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

sum_list( ) → sum([4, 6, 8])

→ 18

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

→ sum([sum_list([1,2,3]), sum_list([4,5]), sum_list(8)])

→ sum([ 6 , 9 , 8 ])

→ 23

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

depth

→ sum([sum_list(1), sum_list([2,2]), sum_list([2,[3,3,3],2])])

→ sum([ 1 , 4 , 13 ])

→ 18

because

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

→ sum([sum_list(1), sum_list([2,2]), sum_list([2,[3,[4,4],3,3],2])])

→ sum([ 1 , 4 , 21 ])

→ 26

10. Are you a believer yet?