

AI

1. Write code for classes that represent game objects (Steps 1-3) in `toah_model.py`.
 - write the provided methods 1st
 - code for GUI shows which methods it expects to interact with
 - don't touch `MoveSequence` class, but use it
2. Write code for a console version `console_controller.py`
3. Write code in `tour.py`. ⇒ Recursive part.

Recursion Exercises

if we're tracing & we find a recursive call we already understand, don't need to trace further.

```
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([1]) sum_list([]) (although sum([])  
    0 is 0 too)  
    """
```

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

→ new-list = []
for x in L:
 L.append(sum_list(x))
return sum(new-list)

1. What helper ^{functions} ~~methods~~ 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])`

→ list of list of ints and int

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

(we've seen what happens when we call `sum_list` on an int, and on a flat list of ints)

→ `sum([4, 6, 8])`

→ 18

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

→ list of list of ints and ints

we already saw how `sum_list` works on lists with just ints + list of ints

→ 23

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

we know how `sum_list` works on all of these.

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

→ `sum([1, 4, 13])`

→ 18

↳ we don't need to trace further

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])])`

↑ we saw this depth here

→ `sum([1, 4, 2113])`

↓

→ ~~24~~ 26

10. Are you a believer yet?