A few more recursion examples
Announcements

1. Lab due tomorrow
   a. Required to write a few recursive functions
   b. How do you identify a recursive function?
   c. Similar to problems shown in class

2. Administrative
   a. QR issue about the test 1 is sorted out and the results will be available soon
   b. I have looked at the first 31, marks are looking good
   c. A1 marks will also be released soon
   d. Demos have been marked already
A word about calling functions from functions

```python
>>> def f(n):
...     return g(n) * 2
... 
... def g(n):
...     return n
... 
>>> f(2)
4
```
Example 5: Count how many items

```python
>>> list_ = ['how', ['now', 'brown'], 'cow']
>>> nested_count(list_)
4
```
Idea

1. We will use `sum()` as the combination function
2. We want to add “1” for each non list element to the argument of `sum` and recursively call the function for all list elements

```python
>>> list_ = ["how", ["now", "brown"], "cow"]
>>> nested_count(list_)
4
```
Questions

1. What inputs will cause no recursion?
Questions

```python
if not isinstance(obj, list):
    return 1
else:
    return sum([nested_count(i) for i in obj])
```

2. Will the code work for empty list?
Recursion with history preserved

1. So far recursions are **blind**
2. They do not know where in the call level it is
   a. Called on a bigger list or a smaller sublist
3. How do we pass the level information
Example 6: List all non-list elements at a level

```python
>>> list_ = [1, [2, [3, 4], 5], 2]
>>> list_level(list_, 2)
[2, 5]
>>> list_level(list_, 3)
[3, 4]
```
Example 7: List all non-list elements at all levels

```python
>>> list_ = [1, [2, [3, 4], 5], 2]
>>> list_levels(list_)
[[[1, 2], [2, 5], [3, 4]]]
Play at home

tree burst (recursion using turtles)