Outline

A1 class design

More recursion

Testing, big and small

Functional Programming
Separation of concerns
Tracing to understand recursion

def remove3s(L: list):
    i = 0
    while i < len(L):
        if isinstance(L[i], int):
            if L[i] == 3:
                del L[i]
                continue
        elif isinstance(L[i], list):
            remove3s(L[i])
        i += 1

Red part is the current value of L.
A relevant example

This is a job for recursion:

\[ M(n) = \begin{cases} 
1 & \text{if } n == 1 \\
\min \{1 \leq i < n \mid 2 \times M(n - i) + 2^i - 1\} & \text{otherwise.}
\end{cases} \]

That’s a recursive formula. Python has a built-in function `min`. You probably want to combine (tuple?) the minimum number of moves with the split \(i\) that produces it.
get some turtles to draw

Spawn some turtles, point them in different directions, get them to draw a little and then spawn again...

Try out tree_burst.py
before and after coding:

Test your docstring examples automatically:

```python
if __name__ == '__main__':
    import doctest
    doctest.testmod()
```

For more thorough testing, use unittest
Nameless functions with lambda

Writing `(lambda x: one-line-function-body)` in a given place in your code accomplishes the same thing as first defining a function

```python
def fn_name(x):
    one-line-function-body
```

and then writing `fn_name` in that same place in your code.

```python
def square(x:int):
    return x**2
print(square(5))
```

```python
print((lambda x: x**2)(5))
```

Nothing deep!
It is simply more-concise and doesn’t require you to introduce a name for the function, which is good if you’re only going to use the function once.
Useful built-in functions to use with lambda

- **map(f, iterable_object)** returns an object of the same type and size as iterable_object obtained by applying the function f to each of iterable_object. What’s this do?

  map(lambda x: x**2, [1, 0, 4, -1])

  You already know this one! Same as
  
  [x**2 for x in [1,0,4,-1]]

- **filter(f, iterable_object)** returns an object of the same type as iterable_object that contains only the elements \( x \in \text{iterable\_object} \) such that \( f(x) \) return true. What’s this do?

  filter(lambda x: x > 0, [1, 0, 4, -1])