

Functions and Fractals: Sierpinski triangles

```

      1
     111
    1 1
   111_111
  1 1
 111_111
1 1 1 1
111_111_111_111
 1 1
 111_111
 1 1 1 1
 111_111_111_111
 1 1 1 1
 111_111_111_111
 1 1 1 1 1 1 1 1
111_111_111_111_111_111_111_111
 1 1
 111_111
 1 1 1 1
 111_111_111_111
 1 1 1 1
 111_111_111_111
 1 1 1 1 1 1 1 1
111_111_111_111_111_111_111_111
 1 1
 111_111
 1 1 1 1
 111_111_111_111
 1 1 1 1
 111_111_111_111
 1 1 1 1 1 1 1 1
111_111_111_111_111_111_111_111
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
111_111_111_111_111_111_111_111_111_111_111_111_111_111_111

```

Sierpinski Triangle

The [Sierpinski Triangle](#) is a pretty fractal which consists of layers of self-similar triangles, nested inside each other. This challenge involves the construction of such triangles, in the form of ASCII Art. The restriction is, that you need to accomplish this with functional programming, and you cannot declare even local variables!

We have to deal with real world constraints, so we cannot keep repeating the pattern infinitely. So, we will provide you a number of iterations, and you need to generate the ASCII version of the Sierpinski Triangle for those many iterations (or, levels of recursion). A few samples are provided below.

Iteration #0

In the beginning, we simply print a triangle which points upwards. There are 32 rows and 63 columns in this matrix. The triangle is composed of underscores and ones as shown below.

```

      1
     111
    11111
   1111111
  111111111
 1111111111
11111111111
111111111111
1111111111111
11111111111111
111111111111111
1111111111111111
11111111111111111
111111111111111111
1111111111111111111

```

Iteration #1

Iteration #2

| |
|-------|
| 1 |
| 111 |
| 11111 |

The Nth triangle of the series shown above. The output will consist of 32 rows and 63 columns, and will be composed of ones (1) and underscores as in the triangles above.