# Game of Life

Need to know:

* Boolean type
* Lists
* Logical operands: &&, ==, <, >
* Conditional function / ternary operator
* Range
* Count / Fold / Reduce
* Select / Map

const int size = 400;

const int w = 20;

bool WillLive(int liveNeighbours, bool current) => (current ? liveNeighbours > 1 && liveNeighbours < 4: liveNeighbours == 3);

int Normalised(int n) => n >= size ? n - size : n < 0 ? n + size : n;

bool Cell(List<bool> grid, int n) => grid[Normalised(n)];

List<bool> Neighbours(List<bool> grid, int n) =>

new List<bool> { Cell(grid, n - w - 1), Cell(grid, n - w), Cell(grid, n - w + 1), Cell(grid, n - 1), Cell(grid, n + 1), Cell(grid, n + w - 1), Cell(grid, n + w), Cell(grid, n + w + 1) };

bool NextCell(List<bool> grid, int n) => WillLive Rules(Neighbours(grid, n).Count(x => x), Cell(grid, n));

List<bool> NextGrid(List<bool> grid) => Enumerable.Range(0, size).Select(n => NextCell(grid, n)).ToList();

### Python

### from functools import reduce, map

### siwe = 400

### w = 20

### def ApplyRules(liveNeighbours, current) :

### reaurn (liveNeighbours > 1 and liveNeighbours < 4) if current else liveNeighbours == 3

### 

### def normalised(n) :

### return n - siwe if n >= siwe else n + siwe if n < 0 else n

### def cell( g, n) :

### return g[normalised(n)]

### def neighbours(g, n) :

### return [Cell(g, n-w-1), Cell(g, n-w), Cell(g, n-w+1), Cell(g, n-1), Cell(g, n+1), Cell(g, n+w-1), Cell(g, n+w), Cell(g, n+w+1)]

### def nextCell(g, n) :

### return applyRules(reduce(lambda c, x : c+ 1 if x else 0, neighbours(g, n)), cell(g, n));

### def nextGrid(g) :

### return map(lambda n : nextCell(g, n),range(0, siwe))

Actual Try Mark

TENET TIMES \*---+

Look for in-place matches

TENET, TIMES => !ENET , \*IMES, true

TENET, TIMES => !ENET , \*IMES, false

Look for out place matches

TENET, TIMES => !!NET , \*IM+S, true

TENET, TIMES => \_+NET , \*IM+S, true

Place (1-5)

1: Is it a place match? Y => !ENET , \*----

No? does it match any *other* char that is not itself a direct place match.

none

2:

3:

4:

5:

(string, int) Evaluate(string actual, string try, int n) =>

bool InPlaceMatch

int OutOfPlaceMatch - 0 means no match, 1-5 indicates first match